CRANFIELD UNIVERSITY

SCHOOL OF MECHANICAL ENGINEERING

PHD THESIS

Academic Year 1999-2000

M. ZEDDA

GAS TURBINE ENGINE AND SENSOR FAULT DIAGNOSIS

Supervisor: Prof. Riti Singh

November 1999

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

ABSTRACT

Substantial economic and even safety related gains can be achieved if effective gas turbine performance analysis is attained. During the development phase, analysis can help understand the effect on the various components and on the overall engine performance of the modifications applied. During usage, analysis plays a major role in the assessment of the health status of the engine. Both condition monitoring of operating engines and pass off tests heavily rely on the analysis.

In spite of its relevance, accurate performance analysis is still difficult to achieve. A major cause of this is measurement uncertainty: gas turbine measurements are affected by noise and biases. The simultaneous presence of engine and sensor faults makes it hard to establish the actual condition of the engine components.

To date, most estimation techniques used to cope with measurement uncertainty are based on Kalman filtering. This classic estimation technique, though, is definitely not effective enough. Typical Kalman filter results can be strongly misleading so that even the application of performance analysis may become questionable. The main engine manufactures, in conjunction with research teams, have devised modified Kalman filter based techniques to overcome the most common drawbacks. Nonetheless, the proposed methods are not able to produce accurate and reliable performance analysis.

In the present work a different approach has been pursued and a novel method developed, which is able to quantify the performance parameter variations expressing the component faults in presence of noise and a significant number of sensor faults. The statistical basis of the method is sound: the only accepted statistical assumption regards the well known measurement noise standard deviations. The technique is based on an optimisation procedure carried out by means of a problem specific, real coded Genetic Algorithm. The optimisation based method enables to concentrate the steady state analysis on the faulty engine component(s). A clear indication is given as to which component(s) is(are) responsible for the loss of performance. The optimisation automatically carries out multiple sensor failure detection, isolation and accommodation. The noise and biases affecting the parameters setting the operating point of the engine are coped with as well.
The technique has been explicitly developed for development engine test bed analysis, where the instrumentation set is usually rather comprehensive. In other diagnostic cases (pass off tests, ground based analysis of on wing engines), though, just few sensors may be present. For these situations, the standard method has been modified to perform multiple operating point analysis, whereby the amount of information is maximised by simultaneous analysis of more than a single test point. Even in this case, the results are very accurate.

In the quest for techniques able to cope with measurement uncertainty, Neural Networks have been considered as well. A novel Auto-Associative Neural Network has been devised, which is able to carry out accurate sensor failure detection and isolation. Advantages and disadvantages of Neural Network-based gas turbine diagnostics have been analysed.

ACKNOWLEDGEMENTS

Working on the subject presented in this thesis has been interesting, exciting and challenging. However, accomplishment of the goal would not have been possible without the help of many people that, in a way or another, contributed to the various stages of the study.

First of all, I wish to express my gratitude to my supervisor, Prof. Riti Singh, who gave me the opportunity to undertake such an interesting work. During these years he has been helpful not only as a mere supervisor by sharing with me his technical knowledge, but also as a guide by helping me with his wisdom and experience. I really appreciate that.

The present work would not have been possible without the interest and support shown by the two sponsoring organisations, Rolls-Royce and DERA. In particular Rolls-Royce have helped development of the work by providing information, guidelines and by showing trust in my capabilities. During the development of the project, I had the opportunity to work with many people from Rolls-Royce (both Derby and Bristol) and their knowledge and helpfulness have impressed me. I especially wish to thank Mr Barry Curnock and Mr Peter Robinson from Bristol, who followed my work since the beginning and gave me useful suggestions. Mr John Martin, Mr Pratap Nayer and Mr John Borrodaile from Derby have to be thanked as well for their interest in the development of the diagnostic method and for their understanding.

The importance of a quiet, friendly and efficient working environment cannot be neglected: I wish to thank the Performance Engineering UTC staff as well, who always supported and helped me.

Eventually, last but not least, I wish to thank all the people who made my stay at Cranfield a pleasant experience. Meeting people from all over the world has definitely been fascinating and interesting. Moreover, I want to express my gratitude to the "Italian community" of Cranfield, which filled my life with fun and friendship.

This long list of acknowledgements must also include those who have been close to me during these years, my parents and my brother. Although far away, their support and encouragement made this experience possible.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

A/C	Aircraft
AANN	Auto-Associative Neural Network
ANN	Artificial Neural Network
AIDS	Aicraft Integrated Data System
AI	Artificial Intelligence
ART	Adaptive Resonance Theory
BP	Backpropagation
BTT	Backpropagation Through Time
CI	Computational Intelligence
CP	Counterpropagation
DERA	Defence Evaluation and Research Agency
DNN	Decentralised Neural Network
EBP	Extended Backpropagation
ECM	Engine Condition Monitoring
EFA	European Fighter Aircraft
EKF	Extended Kalman Filter
EMS	Engine Monitoring System
EP	Evolution Program
ES	Evolution Strategy
ETOPS	Extended range Twin-engine Operations
FADEC	Full Authority Digital Engine Control
FANIN	Fan Inner
FANOUT	Fan Outer
FDI	Failure Detection and Isolation
FIR	Finite-duration Impulse Response
FLS	Fuzzy Logic System
GA	Genetic Algorithms
GE	General Electric
GPA	Gas Path Analysis
HP	High Pressure
ICM	Inflence Coefficient Matrix
IEKF	Iterated Extended Kalman Filter
IP	Intermediate Pressure
KBS	Knowledge-Based System
KF	Kalman Filter
LP	Low Pressure
LPCA	Linear Principal Component Analysis
ML	Maximum Likelihood
MLP	Multi-Layer Perceptron
MME	Minimum Model Error
MNN	Main Neural Netwrok
MNR	Modified Newton-Raphson
MOPA	Multiple Operating Point Analysis
MNRES	Modified Newton-Raphson with Estimated Sensitivities

MV	Minimum Variance
NASA	National Aeronautics and Space Agency
NLPCA	Non-Linear Principal Component Analysis
NN	Neural Network
pdf	Probability Density Function
PEUI	Perfromance Estimation Uncertainty Index
PNN	Probabilistic Neural Network
PSC	Performance Seeking Control
PW	Pratt & Whitney
RAANN	Robust Auto-Associative Neural Network
RR	Roll-Royce
RRAP	Rolls-Royce Aero-engine Performance
RMS	Root Mean Square
SFDI	Sensor Failure Detection and Isolation
SFDIA	Sensor Failure Detection, Isolation and Accommodation
SOAPP	State Of the Art Performance Program
SSME	Space Shuttle Main Engine
SUS	Stochastic Universal Sampling
SVM	State Variable Model
TPBVP	Two Points Boundary Value Problem
WLS	Weighted Least Squares
WRMS	Weighted Root Mean Squares

CHAPTER 1

INTRODUCTION

1.1 Gas turbine performance modelling and analysis

Performance simulation is a useful tool to support design and operation of gas turbines. Simulating the performance of an engine means building a mathematical model able to reproduce with the desired degree of accuracy the physical phenomena occurring in the machine. The model relies on basic aero-thermodynamic principles (conservation of mass and energy, balance of momentum) and component performance maps. To date, most of the modelling is mono-dimensional, even though allowance can be made for two-dimensional effects by means of semi-empirical coefficients, whenever they are available and useful to improve modelling accuracy.
Performance related quantities can be split into two sets:

1) *measurements*: typically they are temperatures, pressures, spools speeds, airflows, fuel flows and thrust. If more than one probe is used to measure pressure or temperature at a certain station, the averaging is weighted either by massflow or by area. The resulting weighted average should represent the corresponding mono-dimensional quantity used by the model.
2) *performance* or *health parameters*: typically they are efficiency and flow function for rotating components and discharge coefficient for propelling nozzles. They quantify the performance of the engine component.

During the preliminary design phase of a gas turbine, models are widely used to help predict the performance of the engine. At this initial stage, simulation enables to analyse several engine configurations quickly and cheaply. Given a certain engine configuration, assignment of the component maps (hence the performance parameters) allows to run the model to predict the overall performance of the proposed solution. The simulation code is said to be run in *synthesis* mode.

During the proper development phase, engine component maps become available and the model is gradually adjusted to better fit the real cycle. At this stage, simulation enables to identify the effects on the overall engine performance of the modifications applied to the components. The input is the set of measurements, the output the set of performance parameters. The simulation code is said to be run in *analysis* mode.
In general analysis can have different objectives depending on what the results will be used for.
Primary aim of development test data analysis is a thorough understanding of the performance of all components, which should support the design process.
When an engine is delivered to the customer or has just undergone a major overhaul, a pass off test is required to assess the achievement or restoration of the desired

performance. At this stage the analysis can actually be considered rather empirical, in that detailed understanding of the performance is not necessary unless the test is failed.
Another major application of analysis occurs in condition monitoring. In this case a fairly detailed level of analysis is required to achieve cost-effective maintenance.

Whereas synthesis is a well-defined technique, analysis is still subject to intensive study to make it effective and reliable. A number of problems, mainly related to measurement uncertainty, hinder the successful application of performance analysis. Most of them are addressed in the present thesis.

1.2 Gas Path Analysis

Analysis of the component performance is required in two main cases. On the one hand the performance of most mechanical devices deteriorates with usage and gas turbines are no exception. The most common cause is the degradation of one or more basic components (compressors and turbines). Fouling, foreign object damage, erosion, corrosion, blade damage, nozzle plugging are just some of the possible physical degradations which can produce reduction of performance. On the other hand during the development phase changes in a component can modify its performance and the overall behaviour of the engine as well.
The component's loss (or even gain) of performance produces a change of the corresponding maps. Even though the health parameters are usually not directly measurable, analysis of the changes induced in the measurements should allow isolating degraded components. When the faulty components are isolated through calculation of the performance parameters' variations, a further step is necessary to relate these changes to physical modifications of the engine components. Fig. 1.1 displays the working principle of performance analysis.

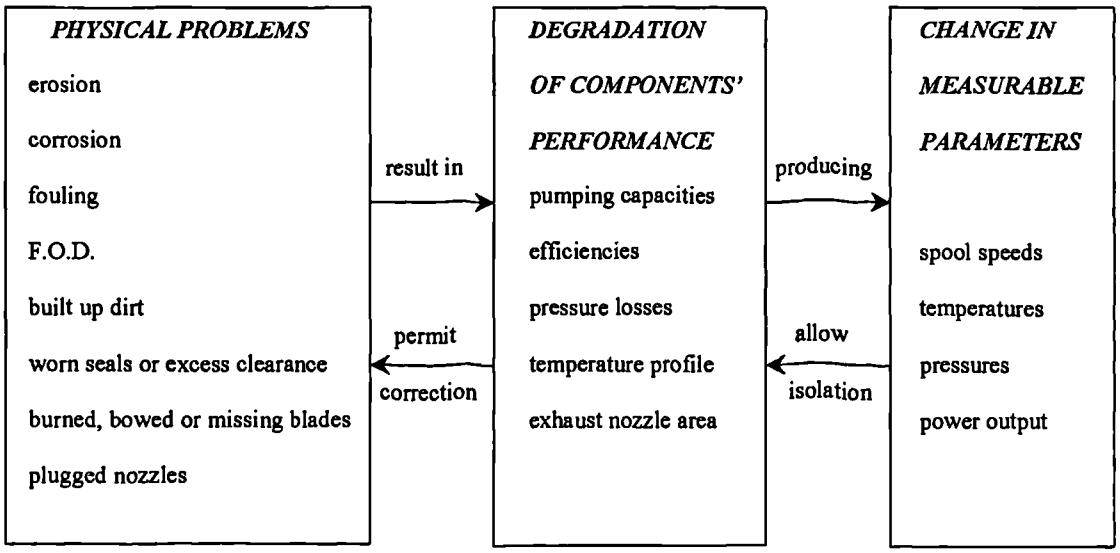


Fig. 1.1: performance analysis

Gas Path Analysis (GPA) is the model-based technique able to calculate the performance parameters by using measurements as input. Steady state aero-thermodynamic equations and component characteristics are used for the purpose. As stated earlier, most of the modelling is mono-dimensional.
If any effect of measurement uncertainty is neglected, for a given engine operating point the basic equation is as follows:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) \tag{1.1}$$

where:

- $\mathbf{z} \in R^M$ is the measurement vector and M is the number of measurements
- $\mathbf{x} \in R^N$ is the performance parameter vector and N is the number of performance parameters
- $\mathbf{h}$ is a vector-valued function, usually non-linear. It is provided by the performance simulation model.

Given the measurements $\mathbf{z}$, the performance parameters $\mathbf{x}$ can be calculated by inverting the function $\mathbf{h}$:

$$\mathbf{x} = \mathbf{h}^{-1}(\mathbf{z}) \tag{1.2}$$

The inversion (1.2) is feasible when the number of performance parameters equals the number of measurements ($N = M$). Otherwise estimation techniques, that will be dealt with in chapter 2, are to be used.

1.3 Engine Condition Monitoring

Deterioration can affect relevant factors such as thrust (or power) and specific fuel consumption. The remarkable influence of component deterioration on the overall engine performance has a thermodynamic explanation: gas turbines are based on the Brayton cycle, whose thermodynamic efficiency is strongly dependent on compressor and turbine efficiency.
As a consequence of progressive performance loss, operation of the engine can become cost ineffective (excessive s.f.c.) or even unsafe (insufficient take off thrust). Therefore maintenance techniques must be used to ensure that the engine is operated cost effectively and safely. Since the introduction of gas turbines, maintenance techniques have been applied. Three main types of maintenance strategies can be identified:

1) *hard time*: the engine and its components are periodically overhauled in accordance to the operator's maintenance manual or are removed from service. The time limitation may be adjusted, based on operating experience or tests as appropriate.
2) *on condition*: the engine and its components are periodically inspected or checked against some appropriate physical limits to determine whether they can continue in service. The purpose is to remove the engine from service before a failure occurs. The physical limits can be adjusted based on operating experience or tests as appropriate.

3) *condition monitoring*: the health status of the engine, inclusive of components and subsystems, is defined through the use of sensor inputs, data collection, analysis and a decision making process.

Relatively recent advances in computing, recording and general modelling capability have made it feasible to develop maintenance strategies mainly based on condition monitoring. Engine Condition Monitoring (ECM) practice is somewhat different for military and civil engines. The main differences are due to:

- available data: steady state data are seldom collected for on wing military engines because of the predominance of transient manoeuvres. Conversely, a large amount of steady state data, collected at various power levels, is available for on wing civil engines.
- primary aims: even though both s.f.c. and thrust are to be monitored and if necessary restored for both civil and military engines, thrust and fuel consumption have a different relative importance in the two cases, because of the stress on fuel consumption for civil and thrust on military engines.

However, cost effectiveness and safety always represent the two key goals.

In today's global civil air transportation market, increasing competition among airlines is pushing towards the application of advanced maintenance techniques to reduce operating costs (Singh et al., 1999). In this respect, the propulsion system calls for a significant portion of the overall maintenance effort. This is amply clear from fig. 1.2, which shows that the engine and maintenance costs together with the fuel bill represent 18% of the total costs. It is also seen that although the profits are large in absolute terms, they are a relatively small percentage of the revenue and costs. Thus any change in either of the two could have a detrimental effect on the total profits. There are certain costs on which the airline has a direct control, whereas others are determined externally. It is these internally affected costs, which mainly include maintenance and to some extent fuel, that can be reduced.

Recently, engine companies have been signing long term aftermarket agreements with airline customers. Instead of the airline repairing and overhauling an engine, the manufacturer does it. The engine company becomes more of a service than a supplier of spare parts. The manufacturer charges the airline for his powerplant usage (i.e. usually by flying hour). This deal is advantageous for the airline, as their cash flows are much more predictable. In the same way, the engine manufacturer has secured more of the revenues associated with the product. This move towards the so-called power by the hour concept will call for long term, comprehensive, advanced monitoring techniques. Furthermore, engine maintenance will be given more emphasis as the airline yields (defined as the revenue generated per passenger-mile) are forecasted to decrease in the next two decades. In fact, the information technology-based yield management systems are unlikely to produce economic gains similar to those experienced when such system entered service. Besides, the relevant initial investment necessary for large commercial turbofan engines, which now dominate the market, requires the development of cost effective maintenance techniques able to restore the engine performance promptly. Eventually, increased reliability and advances in material technology are leading to the move to two powerplants per aircraft, even for extended range operations over water (ETOPS). This fact, coupled with the ever-increasing life on wing for aero-engines, will require parallel advances in monitoring techniques.

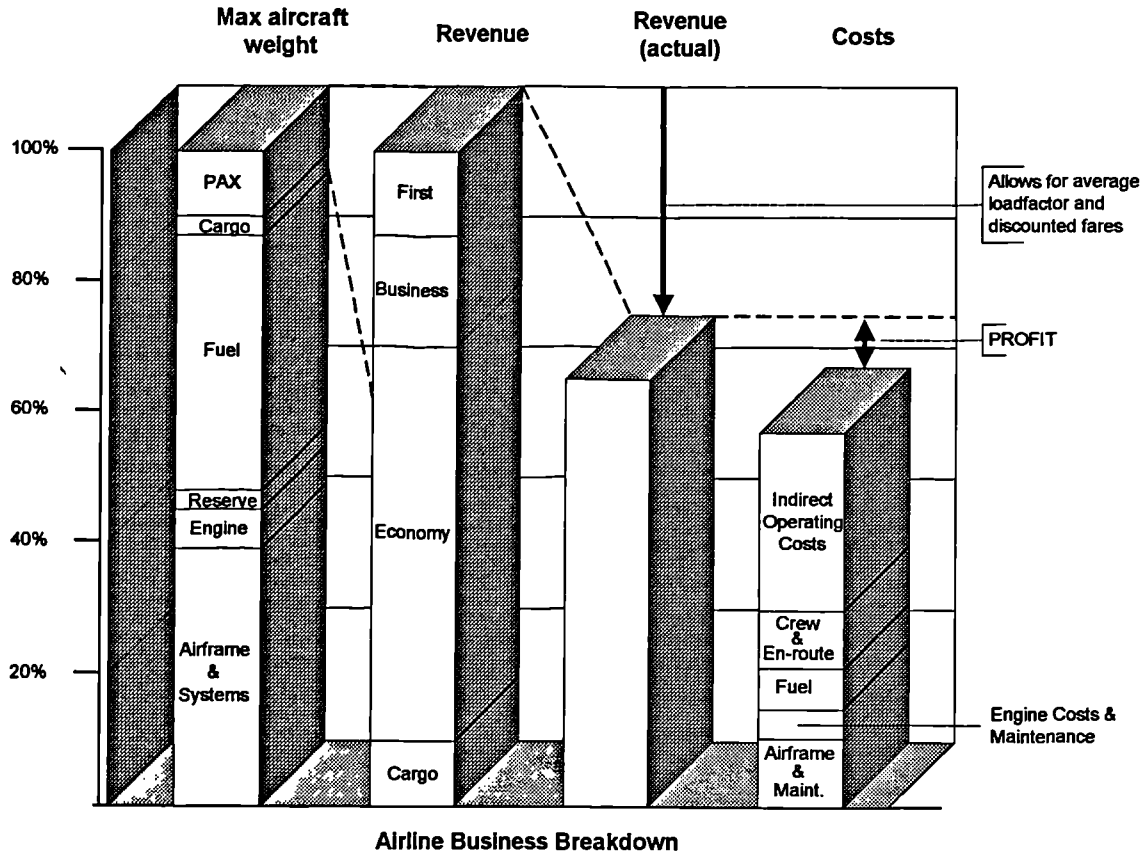


Fig. 1.2: airline business breakdown

Although ECM was first introduced with the commercial application in mind, military engines soon started to be effected by the new maintenance approach: operation and life cycle cost remarkably benefited from the use of ECM techniques. Reasons for the requirement for advanced techniques in the military area are similar to the civil industry, which is to operate as cost-effectively as possible. This is because the budgetary allocations have been reducing over the last few decades, especially with end of the cold war. The outstanding importance of ECM for military engines can be properly appreciated by considering the following example: the eight-engined strategic bomber B52 is planned to remain in service for a long time (2040 is a deadline proposed for the bomber's withdrawn). In the light of the extremely long life on wing in general required for military engines, maintenance will play a major role in the operability and overall life cycle cost of the military aircraft as well.

A key point for any ECM system is the concurrent utilisation of a number of techniques able to keep track of the various components' and subsystems' performance. Gas path analysis, vibration analysis, oil system analysis are just some of the methods to be used. Generally speaking, the techniques can be regarded as complementary in that they detect different features of the on-going faults. Collection and careful processing of the

information provided by the diagnostic subsystems allow drawing a comprehensive picture of the engine's health. An ECM system may encompass the following techniques:

1) aerothermal performance analysis
2) oil analysis
3) vibration analysis
4) visual inspection
5) borescope inspection
6) X ray checks
7) eddy current checks
8) gas path debris analysis
9) usage monitoring
10) turbine exit spread monitoring

Some or all of the techniques listed above (or even more) will be used depending on the complexity and aim of the monitoring system. Even though the present work will focus on aerothermal performance analysis, it would be misleading to assume this technique to be more relevant than the others. Actually these techniques provide a way of monitoring the engine by crosschecking of the results. An interesting case is represented by gas path debris and aerothermal analysis (English, 1995). The former is particularly suitable to detect certain faults at an early stage, while the latter is able to quantify them when they have reached such a level that the component performance is directly affected. From this viewpoint the two techniques can be considered complementary. Another example is the following: for certain faults there exists a link between the features of the Z-modes of rotor vibration and the change in performance parameters of the corresponding components (Carr and Cowley, 1995).

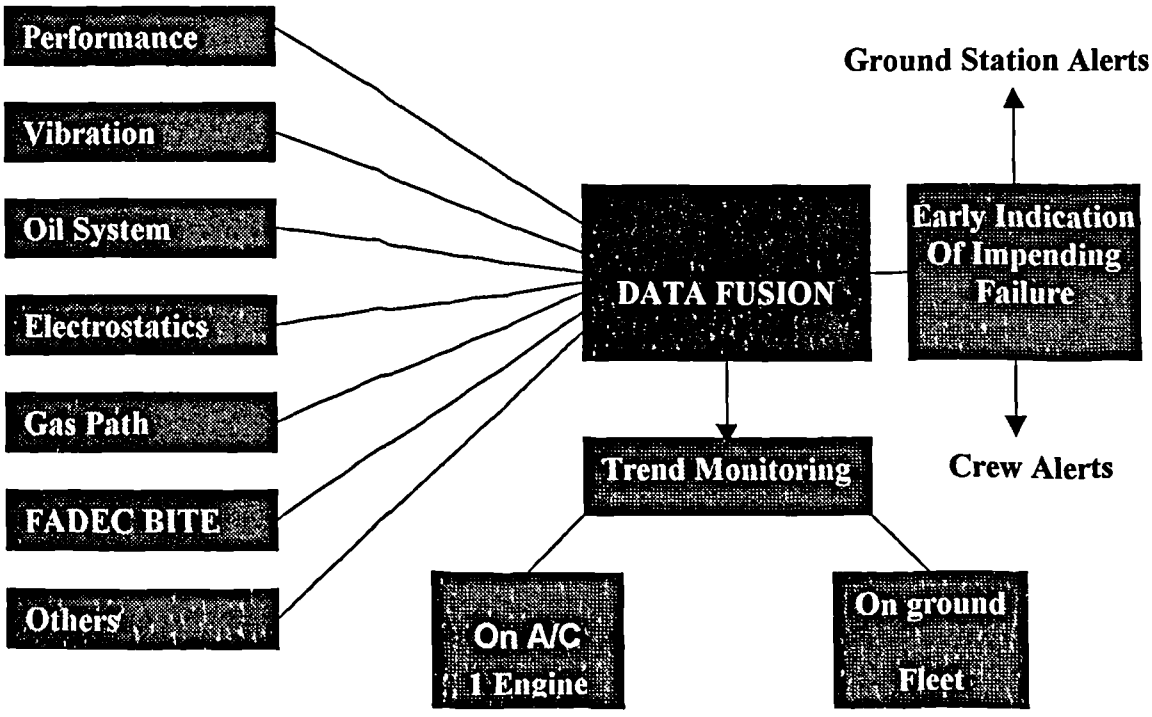


Fig. 1.3: a comprehensive engine monitoring system

As a matter of fact, the current trend is the development of integrated diagnostics able to take all factors into account by means of a probabilistic approach and data fusion techniques (Palmer, 1998; Anuzis, 1998). As stated earlier, ECM systems should rely on a number of different diagnostic methods. To date the analyses are carried out separately and then the results compared to produce a diagnostic answer. However, significant benefits can be achieved by use of data fusion techniques (e.g. Neural Networks, Bayesian Belief Networks, and Knowledge Based Systems) to assess the health status of the engine and its subsystems concurrently (fig. 1.3).

The greatest effort to develop a comprehensive and cost effective monitoring system has been made for aeroengine maintenance. Proper Engine Monitoring Systems (EMSs) have been studied and are currently used. An EMS involves data gathering, processing and recording equipment on the aircraft, supported by substantial amounts of hardware and software equipment in ground-based installations. Moreover, all this equipment is surrounded by a strongly organised group of preferably highly qualified people who can utilise the information generated to support their respective decision making process. In its most general terms, an EMS collects, processes and displays data that can assist engine design, management, safety, operation, maintenance and logistics. An EMS can be manual, computer aided or automated. The introduction of the Aircraft Integrated Data System (AIDS) has positively influenced the EMS for aeroengines. The system is mainly made of the flight data acquisition units, the airborne and the ground based computer. Several parameters are monitored and recorded during flight. Simple limit checking is carried out during flight by the airborne computer and afterwards downloaded data are more thoroughly analysed on the ground.

To assess the consequence of a certain fault or component deterioration, it is necessary to relate it to availability and reliability issues. As a matter of fact, modern engines have very high levels of reliability. When a forced outage occurs, though, availability is affected depending on the down time required to replace or repair the particular component or system.
On the one side, engine support systems (e.g. control and fuel systems) are statistically responsible for a large number of forced outages due to intrinsically low reliability. However, the down time associated is kept to acceptable levels by means of design redundancy, facility of repair/removal and holding of appropriate spares. Moreover, advances in instrumentation and microprocessor-based controllers (e.g. FADEC) can significantly contribute to improve engine support systems' availability.
On the other side, major gas path components such as compressors and turbines have a high reliability. However, when the outage is due to a fault in this kind of components, the down time required to repair the engine is usually very long (see fig. 1.4). The low probability of fault occurrence and the cost of holding engine component spares entail that they are often not held as spares, definitely not in every maintenance platform.

The reliability and availability issues discussed above suggest that relevant gains could be achieved by application of a modular-based performance monitoring technique. For this reason, a great deal of research has been directed to the development of GPA.

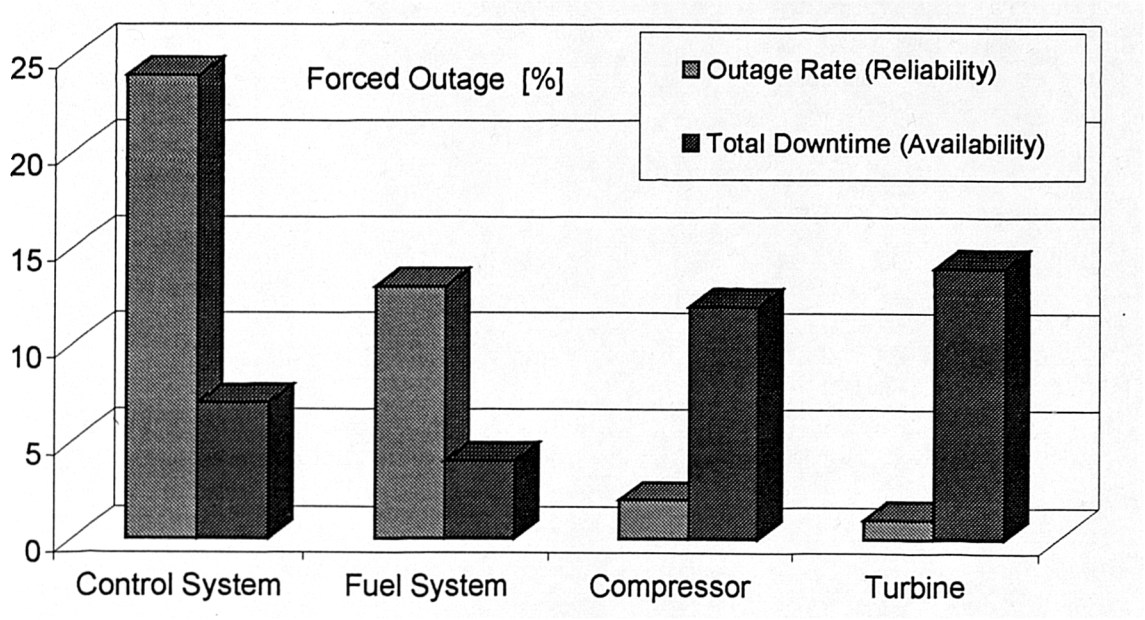


Fig. 1.4: system ranking by forced outage rate and forced outage total down time

Cost effectiveness and safety are the ultimate objectives of any EMS. In more detail, an well-organised EMS can bring benefits in the following areas:

1) flight safety
2) cost saving
3) operation
4) flight deck
5) line and station maintenance
6) depot maintenance
7) logistics
8) engine management.

The fundamental capabilities of an EMS system can be split into three main groups:

1) short term:
- flight safety
- pre-flight status
- in-flight status
- post-flight status
- engine limit exceedances
- lubrication systems
- vibration data

2) intermediate term:
- event analysis
- trim condition
- gas path analysis
- trending
- accessory components

3) long term:
- off-line maintenance information

- data feedback to potential users
- life usage tracking
- mission profile analysis
- record keeping.

Even though ECM principles have been first applied to aeroengines, substantial benefits are possible for industrial gas turbines as well (Zedda et al. 1999). The interest of users of industrial gas turbine engines in condition monitoring is relatively recent. At first, only vibration condition monitoring was performed. The gains achieved with aircraft engine maintenance based on condition monitoring and the widespread use of industrial engines that are actually aero-derivatives have made industrial users more sensitive to maintenance issues. Maintenance systems partially based on condition monitoring have been introduced and are showing the possible economic benefits.

1.4 Objectives and subject matter covered by the thesis

The present work has been fully sponsored by Rolls-Royce plc and DERA. The project objective was twofold:

- a thorough review was required to assess which techniques could be successfully applied to the gas turbine diagnostic problem in its various forms. The methods already known and used have been critically analysed. Pro's and contra's have been highlighted, also in an attempt to identify objectives and methodologies of the various research teams involved in gas turbine diagnostics. New methods have been devised and proposed for a wide range of diagnostic problems. Even though this phase of the study has necessarily been theoretical, in that no implementation was anticipated, whenever possible a critical assessment of advantages and disadvantages of the methods has been provided. In particular, ideas and methods already used in other engineering areas related to process monitoring have been analysed in the light of their possible application to gas turbines
- a system had to be devised, which could carry out engine and sensor fault diagnosis for test bed analysis of development engines. All real-world measurement uncertainty effects, such as noise and biases, had to be coped with properly and effectively. Noise and biases were supposed to affect even the environment and power setting parameters of the engine. A primary feature of the technique had to be the ability to clearly isolate the faulty engine components (and therefore to effect the so-called concentration), as opposed to the typical performance of classic estimation techniques applied to GPA. The focus has been on the diagnostics accuracy. Therefore, the requirement for usage of limited computing power has been regarded as secondary. The diagnostic problem of development engine test bed analysis has been chosen, because of two main reasons: firstly, the design and development phase can substantially benefit from application of accurate and reliable performance analysis; secondly, establishing the maximum level of accuracy achievable when the instrumentation set is comprehensive like in development test beds and when the computing power utilised is a secondary concern allows to set a standard of accuracy and possibly a mainstream technique, from which simplified methods could be derived to tackle other diagnostic problems where the requirements may be different. As a

matter of fact, the last part of the present work dealt with the applicability of the method to pass off tests, where the instrumentation set is more limited. An expansion of the basic technique has been devised and tested, which is able to carry out fault diagnosis with few sensors.

1.5 A guide through the thesis

Chapter 1 is a simple introduction to performance analysis. The utilisation of performance modelling is briefly discussed. A short foray into engine condition monitoring is made, with emphasis on the comprehensive nature of the monitoring process and the economic benefit achievable.

Chapter 2 presents the state of the art of gas turbine performance diagnostics. A detailed definition of Gas Path Analysis is given, along with the main problems encountered in its practical application. The classic approach, based on Kalman filtering, is described under the heading "conventional techniques", together with the modifications proposed by the main engine manufacturers. Advantages and disadvantages of the three methods are analysed. Albeit different, the three methods share similar features, which derive from their common basis, i.e. the Kalman filter. In particular, the study of these well-established methods allowed to clearly identifying the requirements to be satisfied for the proposed diagnostic technique to be useful. Closely related to the classic approach, non-linear Kalman filtering has been analysed and a discussion is presented as to how to deal with the system's non-linearity by means of Kalman filter-based methods. The problem of diagnostics for poorly instrumented engines has been considered as well through multiple operating point techniques. The great deal of work on diagnostics performed by the well-known German team has been described. In particular, it shows how conventional techniques can be complemented by new methods. Although the present work focuses on steady state diagnostics, characteristics and feasibility of transient performance analysis has been considered as well. A novel method, based on eigenstructure assignment within a linear estimation framework, is presented and suggestions are made, as to how it could be expanded. A transient Bayesian diagnostic method is thoroughly studied and a modification is proposed, to make it more suitable to deal with actual measurement noise levels.
A relatively new non-linear estimator (the Minimum Model Error estimator) is proposed to cope with non-linearities and model uncertainties. The method, which has never been considered for gas turbine engine diagnostics before, shows interesting potential, although the necessary computing time may represent a serious limitation in this case. A technique based on an MME estimator has been developed, which is tailored to the problem at hand.
Eventually, a classic estimation technique (maximum likelihood estimator) typically used for identification of aircraft dynamics is modified to better fit the gas turbine diagnostic problem.

Chapter 3 is dedicated to Neural Networks used for diagnostics. A brief introduction to neural techniques is given, together with various applications to process monitoring. Emphasis is put on the choice of the nets and the application to different diagnostic

problems. Sensor Failure Detection, Isolation and Accommodation through Neural Network is then presented. Two different works on Neural Networks applied to gas turbine diagnostics are described. The works represent an attempt to realise what gains can be reached by using advanced, properly designed neural methods. Eventually, a critical examination of the results allows drawing interesting conclusions on the subject of Neural Network-based diagnostics.

Chapter 4 describes the way the diagnostic problem has been solved in the present work. The approach is mainly based on optimisation carried out by means of evolutionary methods. At first the objective function to be optimised is identified, with particular attention paid to measurement noise and biases. Then the various techniques utilised are presented and critically analysed. An introduction to evolutionary-based optimisation methods is given (mainly Genetic Algorithms and Evolution Strategies). The development of the problem-specific Genetic Algorithm is explained and its performance analysed by means of both simulations and theoretical observations. Tests carried out on a two-spool and a three-spool low by pass ratio turbofan engine (EJ200 and RB199) are analysed and discussed in turn. The gains achievable by means of concurrent utilisation of Genetic Algorithms and Evolution Strategies are then shown and related issues analysed.
Eventually, the problem of pass off test data analysis is discussed and the optimisation-based technique is expanded to perform the corresponding diagnostics through simultaneous analysis of more than a single operating point.

Chapter 5 presents final conclusions, comments and recommendations as to the accuracy and applicability of the techniques proposed.

Due to the wealth of methods studied and the analytical nature of the problems, a large number of appendices complement the main text.

CHAPTER 2

ESTIMATION TECHNIQUES FOR GAS TURBINE DIAGNOSTICS

2.1 GPA potential and drawbacks

GPA shows powerful diagnostic potential due to the following capabilities:

- fault's isolation: faulty components are identified by means of calculation of the corresponding performance parameters' variations
- fault's quantification: the fault's severity is directly expressed by the performance parameters. Adequate thresholds can be fixed to reach economic and safe utilisation of the engine
- multiple faults: the technique is able to allow for more than a single engine component fault.

The gains achievable by application of effective GPA to condition monitoring would be the following:

- Spare part provision: the accurate prediction of the usage will enable airlines and overhaul companies to optimise scope and number of spare parts
- Trouble-shooting and repair times can be reduced because of the modular diagnostics provided by GPA. Work can be planned in advance.
- The planning process can be adequately studied to allow repairs to be undertaken at convenient place and time.
- The classic dilemma whether the engine should continue to operate or be removed for major overhaul can be solved by application of accurate and quantitative analysis by GPA.
- Continuity: instead of replacing current on-condition maintenance techniques, GPA can improve and complement them with no need for high initial investment.
- Flexibility: the maintenance process can be tailored for the engine according to its actual building, history and current operation (taking into account mission type, aircraft performance, environment, etc). The flexibility can be reflected in adaptive operational use.
- Component life can be extended: exact knowledge of the engine component conditions allows to safely exploiting possible excess performance to prolong the component's life.

Achievement of the benefits listed above is hindered mainly by the following factors.

*1) **Measurement noise**.* Each measurement is affected by noise and due to the harsh operating environment of gas turbine sensors (high pressure, high temperature, large gradients) the order of magnitude of the noise is often comparable to the variations in the measurements caused by an actual component fault. Neglecting this effect may

completely impair the diagnostics. It is then necessary to modify eq. (1.1), here re-written for convenience:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) \tag{2.1}$$

to account for measurement noise as follows:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{v} \tag{2.2}$$

where $\mathbf{v} \in R^M$ is the zero-mean measurement noise vector.
In this case some sort of estimation technique has to be applied.
2) Sensor faults. Apart from noise, measurements may be further affected by errors constant with time (biases) or even time varying (typically a drift). Although biases can be removed by calibration, the occurrence of sensor faults is rather common and the effect of the consequent measurement error on the diagnostic accuracy can be dramatic. Neglecting a measurement bias can lead to either neglect an actual engine component fault or assume that an actually fault-free component's performance is poor. A correct approach requires modifying eq. (2.2) as follows:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{b} + \mathbf{v} \tag{2.3}$$

where $\mathbf{b}$ is the sensor error. Obviously estimation techniques have to be applied.
It should be noted that the choice of a more accurate sensor will reduce the magnitude of noise $\mathbf{v}$, but the sensor's reliability is likely to worsen. A shorter mean time between failure for the sensor means a higher probability of having systematic or even time-varying measurement errors $\mathbf{b}$.
3) Non-linearity: the relation (2.1) between measurements and performance parameters is highly non-linear.
If measurement uncertainty were neglected, two different approaches would be viable:
a) *linearisation*: this is the classic solution, provided by Urban (1974) for the first time. The function $\mathbf{h}$ is linearised with respect to a chosen point:

$$\Delta\mathbf{z} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_0 \cdot \Delta\mathbf{x} \tag{2.4}$$

The delta quantities are related to the chosen point $(\mathbf{x}_0, \mathbf{z}_0)$ and are often expressed as relative variations:

$$\Delta x_i = \frac{x_i - x_{0i}}{x_{0i}} \tag{2.5}$$

$$\Delta z_i = \frac{z_i - z_{0i}}{z_{0i}} \tag{2.6}$$

Eq. (2.4) can then be written as:

$$\Delta \mathbf{z} = ICM \cdot \Delta \mathbf{x} \tag{2.7}$$

where ICM is the so-called *Influence Coefficient Matrix*. Inverting the ICM produces the required performance parameter vector.
The main advantage of this approach is its simplicity due to the linear form of the derived equations.
b) *iterative process*: no linearisation is carried out and the non-linear vector equation is solved by means of an iterative technique. Eq. (1.2) is reproduced here:

$$\mathbf{x} = \mathbf{h}^{-1}(\mathbf{z}) \tag{2.8}$$

This method ensures a higher diagnostic accuracy provided the iterative process converges.
Handling the non-linearity is straightforward when no measurement uncertainty effects are accounted for. Different iterative methods can be used: Escher (1994) utilises a Newton-Raphson technique, while Sammarco (1988) chooses the Robinson technique.
However, if measurement noise and biases have to be taken into account, non-linear estimation techniques have to be used. That makes solution of the measurement uncertainty related problems harder (Jazwinski, 1770). A diagnostic approach based on optimisation and smoothing should turn out to be more suitable.
4) Number of measurements. A comprehensive assessment of the health of the various gas path components calls for a large number of performance parameters to be evaluated and therefore a large number of sensors to be installed on the engine. If a simple, deterministic approach were pursued by direct calculation of the performance parameters through eq. (2.8), the number of sensors would have to equal the number of performance parameters to satisfy the required balance "equations-unknowns". To a certain extent the application of estimation techniques introduces a greater flexibility in the number of sensors, which do not need to be equal to the number of performance parameters. The estimation's accuracy, though, is strongly dependent on the number of input measurements.
Weight, bulk and cost concerns seriously limit the number of engine's sensors on board of the aircraft; the reliability of the diagnostic system may even be worsened by a large set of sensors, as the probability of a sensor fault increases. As a consequence, on-wing diagnostics often has to be carried out with a small number of sensors (no hardware redundancy) and the goal is to extract the maximum amount of information from the available sensors. As far as test-cell diagnostics is concerned, though, the diagnostic approach can be different due to the availability of a larger number of sensors.
5) Choice of measurements. The instrumentation set should be properly chosen to detect the faults of interest. In fact, given a set of sensors, some faults will be easier to identify than others. Unfortunately, diagnostics usually has to be performed by using the available instrumentation set, which may not be the most suitable because often the choice of sensors is not dictated by GPA accuracy requirements. The system *observability* is a very important issue (Gelb, 1974), even though the development of a quantitative approach to gas turbine engine faults observability has long been neglected (Provost and Singh, 1995).

The measurement uncertainty issues have been tackled in three ways: improvement in sensor design, use of more comprehensive instrumentation sets and development of statistical techniques able to cope with measurement errors.
The improvements in sensor design are steady but relatively slow. Besides, attempts to reduce the non-repeatability ranges by designing more accurate sensors are likely to produce more fragile and then more bias-prone sensors.
Increasing the number of sensors used for monitoring can definitely boost the diagnostic capability. However, a number of issues have to be considered. The instrumentation has its own mean time to failure and needs proper maintenance. Substantial cost is associated with sensor installation and use and in general cost savings are limited by the capability of the technique employed, the design and maturity of the engine, the fuel used, the operating environment and the operating profile. This suggests that an excessive use of instrumentation can result in a loss rather than a gain in terms of Return on Investment (fig. 2.1).

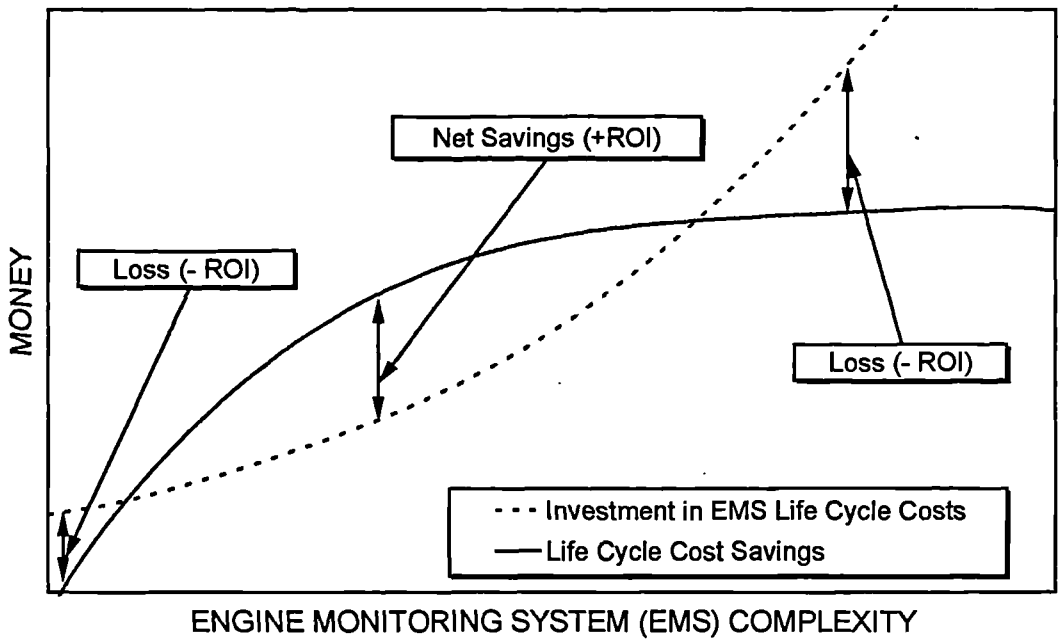


Fig. 2.1: Return On Investment of Engine Monitoring System

Major steps ahead are currently coming from improvements in processing techniques capable to deal with noise and biases. Since Urban's first paper (Urban, 1972), many studies have been developed in order to improve GPA's capabilities. As far as the above issues are concerned, most of the previous and current research developed linearised techniques usually able to cope with measurement noise and in some cases with sensor errors and limitations in the number of sensors. Despite the awareness of the need for methods dealing with measurement uncertainty effectively, for almost two decades classic statistical methods have been used with relatively little success. A widespread technique, named Kalman filtering, has shown its own limitations.

The topic of gas turbine fault diagnosis is of course to be included in the broader area of process fault diagnosis. This means that noticeable advances in fault diagnosis of other kinds of processes may be useful for gas turbines as well. Moreover, gas turbine diagnostics is often linked to the control of the engine, i.e. the same estimation algorithms can be used for both diagnostic and control purposes.
Various techniques have been used or might be used for gas turbine diagnostics. In the sequel brief notes about current and viable approaches are presented. Some improvements or new techniques, which seem to be suitable to tackle particular problems, are theoretically analysed and proposed.

Before embarking upon the literature review, it is worthwhile to make some useful basic distinctions. Aircraft engine health monitoring techniques can be classified as follows:

- ***military engines***: a real-time diagnostics, which can be connected to the control system, is often required. Transient performance is often monitored for diagnostic purposes.
- ***civil engines***: quick on-board diagnostics is often not required, even though it may be desirable.

Another distinction is made between:

- ***on-wing diagnostics***: a set of readings is taken at different points during every flight and from flight to flight for civil engines. The available measurement instrumentation is usually limited and likely to become biased, while the engine is likely to deteriorate with time. The evolution of the degradation is usually unknown, depending on many factors such as the kind of fault, the type of engine and its usage.
- ***test-cell diagnostics***: a larger number of sensors may be installed on the engine and degradation can be assumed to be constant.

A proper diagnostic method has to be able to allow for these differences and therefore be properly tailored.

In the rest of the chapter a review of the most advanced works on gas turbine diagnostics is given. It is worthwhile to point out that many diagnostic techniques have been analysed in that they could provide very useful hints on how to develop a problem specific diagnostic system, even though the application was somewhat different. For instance both steady state and transient diagnostics have been studied, even though the present work focuses on steady state.
Interesting modifications to the existing methods have been developed and assessed on a theoretical basis. Mention of the novel features of the proposed techniques is given along with a brief list of pro and contra.

2.2 Conventional estimation techniques applied to gas turbine diagnostics

Due to measurement noise (point 1 above), sensor errors (point 2 above) and the small number of measurements usually available (point 4 above), a straightforward application of GPA usually produces inaccurate diagnostics. Therefore, many analytical techniques have so far been used in order to provide an estimation of the health of the engine components in presence of such uncertainties. Most estimation techniques have been

applied to linear (or linearised) problems, in that the non-linearity (point 3 above) makes the problem of estimation hard to solve.
If linear GPA is chosen, classic smoothing and filtering methods are available. Among them, the *Kalman Filter* (KF) is particularly effective and has been used by all the gas turbines' main manufacturers (General Electric, Pratt&Whitney, Rolls-Royce) in some form. The present review is based on the current published papers on the subject and shows that the different approaches pursued actually show similarities.

2.2.1 Kalman filter and weighted least squares

The estimation technique described here is applied to a discrete process defined by the following sets of equations:

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{v}_k \qquad \text{measurement equation} \qquad (2.9)$$

$$\mathbf{x}_k = \Phi_{k-1}\mathbf{x}_{k-1} + \mathbf{w}_{k-1} \qquad k = 1,2,\ldots \quad \text{system equation} \qquad (2.10)$$

where:

- $\mathbf{z}_k \in R^M$: measurement vector
- $\mathbf{x}_k \in R^N$: state vector
- $H_k \in R^{MxN}$: model matrix
- $\mathbf{v}_k \in R^M$: measurement noise, assumed to be Gaussian, white (i.e. uncorrelated), zero-mean, with covariance matrix R_k
- $\Phi_k \in R^{NxN}$: transition matrix
- $\mathbf{w}_k \in R^N$: process noise, assumed to be Gaussian, white, zero-mean, with covariance matrix Q_k.

Matrices H_k and Φ_k and measurement and process noise statistics are assumed to be known. The following initial conditions are assumed:

$$E[\mathbf{x}(0)] = \hat{\mathbf{x}}_0 \qquad (2.11)$$

$$E[(\mathbf{x}(0) - \hat{\mathbf{x}}_0)\cdot(\mathbf{x}(0) - \hat{\mathbf{x}}_0)^T] = P_0 \qquad (2.12)$$

where the operator $E[.]$ is the mean value.
Another assumption is that process and measurement noises are uncorrelated:

$$E[\mathbf{w}_i \cdot \mathbf{v}_j] = 0 \qquad \text{for all } i,j \qquad (2.13)$$

The Kalman filter produces a recursive estimation $\hat{\mathbf{x}}_k$ of the state vector at time k based on the current measurement vector $\mathbf{z}_k$ and the previous state vector estimation $\hat{\mathbf{x}}_{k-1}$. This feature can be exploited for real-time applications (quick and simple computations with little memory requirements).
Provided the above hypotheses are all correct, the Kalman filter provides the *minimum variance, unbiased* and *consistent* estimate of the state vector, given a set of measurement vectors.
The estimate is minimum variance as it minimises the following quantity:

$$J_k = E[\tilde{\mathbf{x}}_k^T(+)\cdot\tilde{\mathbf{x}}_k(+)] \qquad (2.14)$$

where:

- $\tilde{\mathbf{x}}_k$ is the estimation error: $\tilde{\mathbf{x}}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k$
- (+) means that the quantity has been updated with the measurement vector $\mathbf{z}_k$
- (−) means that the quantity has been evaluated just before the measurement.

An unbiased estimation is one whose expected value is the same as the quantity to be evaluated:

$$E[\hat{\mathbf{x}}_k] = \mathbf{x}_k \qquad (2.15)$$

A consistent estimate is one which converges to the true value of $\mathbf{x}$ as the number of measurements increases.
The following equations make up the Kalman filter (Gelb, 1974):

$$\hat{\mathbf{x}}_k(-) = \Phi_{k-1}\cdot\hat{\mathbf{x}}_{k-1}(+) \quad \text{state estimate extrapolation} \qquad (2.16)$$

$$P_k(-) = \Phi_{k-1}P_{k-1}(+)\Phi_{k-1}^T + Q_{k-1} \quad \text{error covariance extrapolation} \qquad (2.17)$$

$$\hat{\mathbf{x}}_k(+) = \hat{\mathbf{x}}_k(-) + K_k(\mathbf{z}_k - H_k\cdot\hat{\mathbf{x}}_k(-)) \quad \text{state estimate update} \qquad (2.18)$$

$$P_k(+) = (I - K_kH_k)P_k(-) \quad \text{error covariance update} \qquad (2.19)$$

$$K_k = P_k(-)H_k^T(H_kP_K(-)H_k^T + R_k)^{-1} \quad \text{Kalman gain matrix} \qquad (2.20)$$

As shown by eq. (2.14), the KF minimises a quadratic cost function step by step, i.e. after each measurement. It can be shown that if the system is linear as given by eq. (2.9) and (2.10) then minimisation of (2.14) over the time steps is the same as the minimisation of the complete cost functional evaluated over the whole set of measurements, according to the *weighted least squares* (WLS) technique:

$$J = \frac{1}{2}(\mathbf{x}(0) - \hat{\mathbf{x}}_0)^T \cdot P_0^{-1} \cdot (\mathbf{x}(0) - \mathbf{x}_0) + \frac{1}{2}\sum_{i=1}^{k-1} \mathbf{w}_i^T Q_i \mathbf{w}_i + \\ + \frac{1}{2}\sum_{i=1}^{k} (\mathbf{z}_i - H_i \mathbf{x}_i)^T \cdot R_i^{-1} \cdot (\mathbf{z}_i - H_i \mathbf{x}_i) \tag{2.21}$$

It can be shown (Bryson and Ho, 1975) that for a linear system described by equations (2.9) and (2.10) and subject to the above assumptions the solution provided by the Kalman filter is optimal with respect to every common criterion (minimum variance, maximum likelihood, minimum error). Moreover the technique is recursive.
The strong link between KF and WLS techniques should be noted.

The reason why KF- and WLS-based estimation techniques have been applied to linear GPA is that they seem to show the following advantages:

1. ***optimality***: the cost functional is minimised
2. ***recursivity***: memory and computing requirements are limited
3. ***prior knowledge***: knowledge about the statistics of engine components deterioration can be introduced through the initial values of the state vector and its covariance matrix
4. ***measurement noise***: the actual measurement noise can be assumed to be white and Gaussian, as the Kalman filter requires
5. ***sensor errors***: they can be estimated through augmentation of the state vector to include the unknown sensor biases.

However, real drawbacks in the application of KF techniques to linear GPA are:

1. ***prior knowledge and tuning***: the choice of the process noise covariance matrix (the so-called *tuning*) is often arbitrary. There usually exists no statistically significant population of faulty engines to base the performance parameter's standard deviations' assignment on. Sensitivity studies can be helpful but the dependence of the final diagnostic answer on the chosen standard deviations of the state vector elements may be strong. This means that the use of prior knowledge about the possible deterioration's statistics may produce low-accuracy diagnostics because it is incomplete or incorrect. Due to the artificial introduction of process noise and its covariance matrix, no claim can be made concerning the optimality of the resulting filter.
2. ***"smearing" effect***: often only a limited number of components and sensors are fault-affected, while the KF tends to "smear" the faults over a large number of engine components and sensors. This is due to two main reasons. Firstly, use of the input measurements to estimate both measurement biases and engine component faults makes the problem highly underdetermined. Secondly, the solution given by the KF is a maximum likelihood one and an estimated state vector with a large number of fault-affected elements is more likely than an estimated state vector with a smaller number of fault-affected elements. *Concentration* on the faulty engine components may be difficult as well as decision making as to the action to be undertaken to restore engine performance.
3. ***system model and divergence***: the Kalman filter produces an optimal solution provided the hypotheses about the system are correct. In the case of gas turbine diagnostics, even though we might assume equation (2.9) to be sufficiently precise,

almost nothing is known about equation (2.10), which describes the temporal evolution of the fault. As the method should be able to detect deterioration due to various kinds of faults, both slowly varying (erosion, corrosion, fouling) and abruptly varying (foreign object damage), equation (2.10) is not available. Therefore, it should be somehow estimated and this can impair the final diagnostic accuracy. In fact the use of techniques to completely estimate equation (2.10) introduce errors and as measurements are collected and used by the algorithm "the system learns the wrong state too well". The consequence is *divergence*, i.e. the estimated solution becomes more and more distant from the actual solution.

4. ***Non-linearity and optimality***: the errors due to approximation of a non-linear system with a linear one may not be negligible even if no estimation technique is employed (Escher, 1995; Singh and Escher, 1995). Therefore a non-linear estimation technique seems more suitable. The application of non-linear versions of the Kalman filter, though, is no easy task. Many problems are associated with the use of the common Extended Kalman Filter (EKF) and the Iterated Extended Kalman Filter (IEKF), as pointed out by Jazwinski (1970) and Haupt et al. (1995). The main drawbacks are that the estimates are often biased and suboptimal (i.e. the cost functional is not minimised).

It should eventually be reminded that a correct assessment of the performance of an estimation algorithm has to evaluate the behaviour of the method in case of wrong or incomplete modelling (robustness).
Even though identification of the factors underlying the poor performance of KF techniques applied to linear GPA has not always been thorough, the application has soon shown that accuracy and reliability were not good enough for diagnostic purposes. For this reason, the three main manufacturers have devised modified KF based techniques to cope with the issues described above. In the sequel these are briefly shown, together with other work available in public literature on gas turbine performance analysis.

2.2.2 The Rolls-Royce approach

Rolls-Royce approach has been based on the use of a modified version of the Kalman filter (the so-called "Concentrator") applied to linear GPA (Barwell, 1987; Provost, 1988, 1995). The method has been developed for ground-based diagnostics of aero-engines and can be used for both on-wing and test-cell fault diagnosis. The following section is directly based on the work by Provost (1995).
A correct diagnostic approach has to take account of both component changes and sensor biases. Biases may affect both sensors used to define the environment and power setting condition and sensors used to detect possible faults on the engine (monitoring measurements). Therefore the ICM can be properly modified: if P measurements are used to define the environment and power setting conditions, M is the number of monitoring measurements and N is the number of performance parameters to be calculated, the $(M \times (M+N+P))$ system matrix, shown in fig. 2.2, will be obtained.

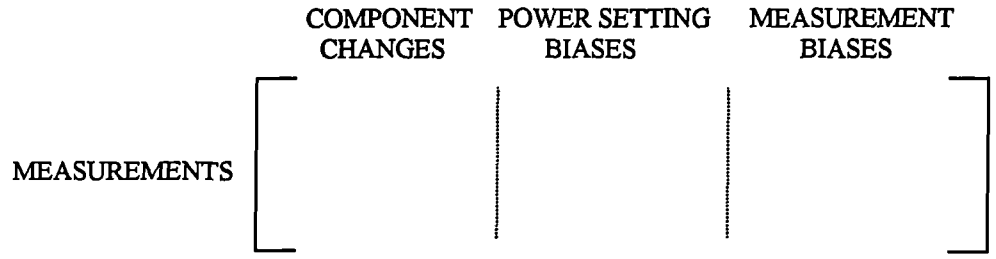


Fig. 2.2: the system matrix

Using this kind of system matrix means to augment the state vector to include not only performance parameters ($\mathbf{x}$) but also measurement biases ($\mathbf{b}$). Both measurement and performance parameters are expressed as delta values according to (2.5) and (2.6). The notation "Δ" is here dropped for convenience.
The "smearing" effect is reduced by means of a concentration technique which makes it possible to focus on a small number of component changes and/or sensor biases,
For test-cell diagnostics the following steps have to be made:

1. the common KF is run
2. the following objective function is evaluated:

$$f = (\mathbf{z} - H\hat{\mathbf{x}})^T \cdot R^{-1} \cdot (\mathbf{z} - H\hat{\mathbf{x}}) \tag{2.22}$$

   where R is the measurement noise covariance matrix.
3. each element of the state vector $\hat{\mathbf{x}}$ is normalised by dividing by the square root of the respective diagonal element of the estimation error covariance matrix:

$$\hat{x}_{ni} = \frac{\hat{x}_i}{\sqrt{P_{ii}}} \tag{2.23}$$

   where P is the estimation error covariance matrix.
4. to start the concentrator, those elements of the state vector are removed which have an absolute value less than the corresponding average; once the concentrator has started, those elements are removed which reduce as the concentrator proceeds
5. the KF is re-run, including only those state vector elements not removed by the previous step
6. steps 2 through 5 are repeated until only one component change and/or sensor bias is present
7. a maximum value for the objective function is selected, either by the user or by the Chi-square probability distribution function (Spiegel, 1972). The vector made up by the smallest number of elements able to produce an objective function less than the selected value is the proposed solution
8. rejecting elements having absolute levels less than a user specified number of standard deviations further modifies the above solution.

For on-wing diagnostics the above algorithm is modified according to the following rules:

1. a system equation, such as eq. (2.10), would be necessary. As a relationship of this kind is not available for gas turbine deterioration, its form has to be estimated as well. The proposed system model equation is:

$$\mathbf{x}_k = (\mathbf{x}_{k-1} + \mathbf{w}_{1k-1}) + t \cdot (\dot{\mathbf{x}}_{k-1} + \mathbf{w}_{2k-1}) \tag{2.24}$$

$$\dot{\mathbf{x}}_k = \dot{\mathbf{x}}_{k-1} + \mathbf{w}_{2k-1} \tag{2.25}$$

2. each measurement time series is processed by KF with the system model equations given by (2.24) and (2.25).
3. the concentrator is applied to each point of the series.

The main advantage of the "pre-filtering" technique chosen is that noise is filtered out before the state vector estimation is performed.

Many techniques can be used to tune the KF, i.e. to assign standard deviations to component changes and sensor biases (engineering judgement, measurement scatter, analysis scatter and mean cost functional technique). Even though engineering judgement is deemed to be the most practical and often effective way of tuning, it is desirable to adjust the standard deviations to allow for the fact that some component changes and sensor biases will be more easily detectable than others.

The product KH is made of a set of columns of best estimates of component changes and sensor biases from the analysis of measurement differences due to 1% change in each component change or sensor bias being searched for. A correct result is not retrieved because the number of unknowns is larger than the number of equations at hand. Each column is divided by the square root of the appropriate diagonal element of the estimation error covariance matrix to get $\frac{KH}{p}$. It is desirable to get a column where the element corresponding to the component change or sensor bias being analysed is the largest. Therefore, the following quantity is introduced: $\frac{KH}{pr}$ where r is the normalised component change or sensor bias being analysed. A sensitivity study can be carried out by calculating the following partial derivative:

$$\frac{\partial\left(\frac{KH}{pr}\right)}{\partial s} \tag{2.26}$$

where s is the standard deviation of the component change or sensor bias considered.

The use of heuristics requires the following relation to be applied:

$$\frac{\text{required change in ratio}}{\text{change in standard deviation}} \cong \frac{\partial\left(\frac{KH}{pr}\right)}{\partial s} \tag{2.27}$$

So the change in standard deviation is obtained:

$$\text{change in standard deviation} \cong \frac{\text{required change in ratio}}{\dfrac{\partial\left(\dfrac{KH}{pr}\right)}{\partial s}} \tag{2.28}$$

Another way of tuning the algorithm is through optimisation techniques, because the real aim is to make the matrix KH as close to the identity matrix as possible. In this problem the inputs are the elements of the process noise covariance matrix (Q and the cost functional is the sum of the absolute values of the off-diagonal elements of the KH matrix). The downhill Simplex method of Nelder and Mead is used (Press et al. 1992). Whereas the first technique is easier to grasp, the optimisation method provides better results and moreover an easier mechanisation of the algorithm.

The RR method definitely represents a step ahead in the quest for estimation techniques suitable for gas turbine diagnostics, especially for the comprehensive approach accounting for both measurement uncertainty and concentration issues. However, the following pitfalls can be highlighted:

- Even though attention is paid to the issue of tuning, the results can still be biased, as the problem of the statistically poor information on engine faults is not overcome.
- Non-linear effects are completely neglected. Accuracy is likely to be affected because gas turbines perform in a remarkably non-linear way and moreover the filter is inherently not robust with respect to unmodelled effects.
- The accuracy of the concentrator has turned out to be poor, as highlighted in a RR technical report (Curnock, 1995) where the method has been used to analyse RB199 test bed data. The overall rate of success in the identification of the faulty engine components has been about 60%.
- The capability to cope with a relatively large number of measurement biases is limited.

2.2.3 The General Electric approach

General Electric approach has been based on the use of Weighted Least Squares applied to linear GPA (Doel, 1994a, 1994b). A computer program (TEMPER) has been developed for both on-wing and test-cell diagnostics of civil engines. Both component changes and sensor biases are estimated.
The usual assumptions already described in the section about KF are used and for a snapshot calculation the cost function (2.21) reduces to:

$$J = \frac{1}{2}(\mathbf{x}(0) - \hat{\mathbf{x}}_0)^T P_0^{-1}(\mathbf{x}(0) - \hat{\mathbf{x}}_0) + \frac{1}{2}(\mathbf{z} - H\mathbf{x})^T R^{-1}(\mathbf{z} - H\mathbf{x}) \tag{2.29}$$

Minimisation of J with respect to $\mathbf{x}$ leads to:

$$\hat{\mathbf{x}} = (P_0^{-1} + H^T R^{-1} H)^{-1} H^T R^{-1} \cdot \mathbf{z} \tag{2.30}$$

A large sample of data can be used to set constraints on the standard deviations. The following equation is used:

$$Q = R + HP_0H^T \tag{2.31}$$

Where the notation is as usual. Eq. (2.31) sets limits on R and M, but does not uniquely define them.

It can be shown (Bryson and Ho, 1975) that the expectation of J when calculated for $\hat{\mathbf{x}}$ is $M/2$, where M is the number of independent measurements. The actual value of $J(\hat{\mathbf{x}})$ will depend on how well the model is able to fit the observed data. As $J(\hat{\mathbf{x}})$ increases the assumed statistical model is less likely to be correct for the particular data sample. $J(\hat{\mathbf{x}})$ is assumed to follow the Chi-squared distribution (Spiegel, 1972) and thereby when its value exceeds the 95% confidence limit the result provided by (2.30) is rejected. Therefore this property of $J(\hat{\mathbf{x}})$ is used to modify the standard WLS technique in order to reduce the effect of smearing typical of this kind of algorithms. Whereas RR technique relies on the concentrator to focus on the most likely faults, GE technique analyses component changes and sensor errors one by one by increasing the standard deviation by a factor 100. This means that the uncertainty about the chosen quantity is increased. If the large $J(\hat{\mathbf{x}})$ is due to a fault in that sensor or engine component, the new analysis should produce a much lower value. The lowest $J(\hat{\mathbf{x}})$ suggests the solution. This modification of the WLS technique is named *fault logic*.

It should be noted that the fault logic technique is able to detect a large fault present only in a single sensor or component: no multi-fault capability is provided in case of large deterioration. If this is not a problem for sensor errors, it may be a problem for the assessment of component faults because a physical fault often produces variations of more than one performance parameter even for a fault located in a single engine component (e.g. compressor efficiency and flow capacity variation due to fouling). However, the system provides the two more likely faults and the user has to decide what is happening.

On this basis both test-cell and on-wing diagnostics are developed.

In test-cell diagnostics measurement baselines are used to represent the performance exhibited by a typical overhauled engine and then the fault logic-augmented WLS technique is applied.

For on-wing diagnostics an equation similar to (2.10), describing the variation of the state vector with time, is necessary. Rather than using a proper KF, exponential smoothing is utilised:

$$\chi_i = \chi_{i-1} + \alpha(x_i - \chi_i) \tag{2.32}$$

where:

- χ_i is the smoothed value at the i-th time step
- x_i is the unsmoothed value at the i-th time step
- α is the smoothing coefficient

The choice of α is a matter of tuning.

A disadvantage of the exponential smoothing with respect to the KF approach proposed by RR is that time lag between the observations and the estimates is likely to occur. However, exponential smoothing is easier to implement

The initial baseline for on-wing diagnostics is derived from the observed performance of newly installed production engines. The first ten valid readings are used to modify the fleet baseline for the specific engine. Then the a priori estimate for each new reading is set by the running baseline from the previous reading. If the fault logic identifies some problem, the engine component or sensor responsible for the mismatching is not subject to smoothing, which is applied to the other elements of the state vector. However, if no way of reducing the $J(\hat{\mathbf{x}})$ is found, there is no updating.

Doel (1994b) presents a comprehensive assessment of the use of WLS techniques by GE. The main points are the following:

- modifications of the standard statistical techniques based on KF (or equivalently WLS) are necessary to reduce the "smearing" effect. It should moreover be noted that diagnostic reliability and accuracy are required especially when large deteriorations are present: the condition of the engine is much more critical in those cases
- different methods have to be developed for on-wing and test-cell diagnostics
- baselines and other statistical inputs require extensive data analysis and careful judgement
- observability of the faults can only be improved by a careful choice of the sensors to be added
- single element sensors cannot be expected to respond consistently to the variety of mechanisms causing component deterioration
- redundant measurements are needed to diagnose sensor faults (hardware redundancy)
- sensors design and placement have to be carefully studied to provide a more reliable indication of the plane average pressures and temperatures, otherwise baseline generation will continue to limit the effectiveness of the algorithms, especially for on-wing diagnostics
- the interpretation of results produced by the WLS requires thorough knowledge of the way the algorithm works. Neural networks, fuzzy logic and model-based reasoning might be suitable tools for augmenting the algorithm and ease interface with the user
- one possible improvement would be the simultaneous analysis of the two readings normally acquired in the acceptance run. They refer to take-off and maximum continuous power setting (the difference in thrust being about 10%). Assuming that the engine component performance parameters are unchanged in the two close power settings, while the measurements change, further information might be extracted for the estimation (Stamatis and Papaliou, 1988; Stamatis et al., 1989). Even though efficiencies and flow capacities are likely to change with operating condition due to aerodynamic reasons, these changes are probably small because the power settings are close to each other. The use of this technique should eliminate inconsistencies between the two readings and improve the analysis of highly deteriorated components. See section 2.3 for details.
- another possible improvement would be the introduction of deterioration models able to describe the variation with time of the expected values of the state vector elements

and of their standard deviations. Basically, this modification would introduce equations similar to (2.10).

In conclusion, GE's point of view is that the next major advance must come from sensor designers and that without advance in sensor technology further modification of the algorithm can only produce small gains.

2.2.4 The Pratt&Whitney approach

Pratt&Whitney, through Hamilton Standards, have been deeply involved in the development of gas turbine diagnostics via a KF based method. Several modifications have been applied to the standard KF to cope with some of the filter's drawbacks (Urban and Volponi, 1992; Volponi, 1994).
The most relevant features of the proposed diagnostic method are the following:

- measurement biases are estimated by augmenting the state variable vector:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ \mathbf{b} \end{bmatrix} \tag{2.33}$$

Thus the measurement equation (2.9) becomes:

$$\mathbf{z}_k = H_{ek} \cdot \mathbf{x}_k + H_{sk} \cdot \mathbf{b}_k + \mathbf{v}_k = H_k \cdot \mathbf{X}_k + \mathbf{v}_k \tag{2.34}$$

where H_k is partitioned as follows:

$$H_k = [H_{ek} \vdots H_{sk}] \tag{2.35}$$

in order to distinguish between engine and sensor faults.
Both performance parameters $\mathbf{x}$ and biases $\mathbf{b}$ are estimated.

- straight application of the KF can actually lead to numerical instabilities and ultimately divergence, as expected from a simple theoretical assessment of the filter's capability. The problem can be overcome by imposing the following conditions:

$$Q_k = 0 \tag{2.36}$$

$$P_k = P_0 = const \tag{2.37}$$

Condition (2.36) entails there is no process noise, while condition (2.37) assumes the error covariance matrix to be constant. Thus, a modified KF is obtained, where the error covariance extrapolation (2.17) and the error covariance update (2.19) are eliminated.

- As to the transition matrix Φ_k, two strategies can be pursued:
  1. No knowledge of the fault evolution is assumed:

$$\Phi_k = I \tag{2.38}$$

where I is the identity matrix

2. Φ_k is approximated for various classes of engines and is a function of the flight cycles.

The points to be made in this respect are:

- the gains achieved by use of these approximate deterioration evolution models are small, although the information about the direction of the changes can be helpful
- the filter works well only when the deterioration model is very accurate. However, the model itself is necessarily a rough approximation. Thus, in general the estimation accuracy is poor, especially for new kind of faults and engines
- no guess can be taken as to the likely evolution of sensor faults.

- For a snapshot analysis when no a priori information is available, the estimation error is related to the eigenvalues of the matrix $(I-KH)(I-KH)^T$. In particular, the average squared error of estimation, normalised by the augmented vector $\mathbf{X}$, is a weighted average of the eigenvalues as shown below:

$$\frac{|\tilde{\mathbf{X}}|^2}{|\mathbf{X}|^2} = \frac{\sum_{i=1}^{N+M} A_i^2 \lambda_i}{\sum_{i=1}^{N+M} A_i^2} = \sum_{i=1}^{N+M} w_i \lambda_i \tag{2.39}$$

Bounds for the average normalised error result as follows:

$$\sqrt{\lambda_{\min}} < \frac{|\tilde{\mathbf{X}}|}{|\mathbf{X}|} < \sqrt{\lambda_{\max}} \tag{2.40}$$

The bounds provided by (2.40) refer to the average error, the actual error being slightly more or less depending on the signature of the measurement noise. The estimation of the performance parameter deltas $\mathbf{x}$ and measurement biases $\mathbf{b}$ (hence $\mathbf{X}$) corresponding to the eigenvector associated with the smallest eigenvalue $\lambda_{\min}$ should be affected by the smallest error.

- A large measurement error correction algorithm enhances the filter. Single measurement biases can be detected, isolated and accommodated. The underlying idea to distinguish between a single measurement bias and an engine fault is that the former will influence just one parameter (the measurement bias itself) while the latter will cause changes in several measurements. Appendix A contains the mathematics. The main outlines of the methods are:
  - For every bias a threshold is defined. Whenever an exceedance is observed, the accommodation process is triggered
  - Upper and lower bounds for the values of the thresholds can be defined (see appendix A)

- Whenever a faulty sensor is isolated (e.g. the j-th), the corresponding bias is evaluated as follows:

$$\hat{b}_j = \bar{b}_j + \frac{[KH_s]_j \cdot (\hat{\mathbf{x}} - \bar{\mathbf{x}})}{\left|[KH_s]_j\right|^2} \tag{2.41}$$

where:
  - $\bar{b}_j$ is the value of the bias' a priori estimation
  - $\bar{\mathbf{x}}$ is the value of the performance parameter vector's a priori estimation
  - $\hat{\mathbf{x}}$ is the current estimation of the performance parameter delta vector.
- Obviously the value of $\hat{b}_j$ does not depend on the value of the corresponding measurement, because it is estimated by using the other measurements which are supposed to be bias-free.

The following points have to be made:
- Poor information on the fault evolution leads to inaccurate results and even divergence.
- Even though an iterative technique is suggested to cope with biases affecting the environment and power setting parameters, the claimed accuracy is low and a limit on the number of iterations of the method has to be imposed in order to avoid divergence problems.
- The recovery algorithm for measurement biases works with just one faulty sensor.
- The $N+M$ state variables are estimated by using M input measurements: even in the bias-free case inaccuracy is likely to affect the results simply due to "smearing". No concentration on the faulty engine components is attained.
- The solution is not optimal in the sense of minimisation of the function (2.21). The consequence is inaccuracy.

A brief analysis of the typical results confirms the drawbacks of KF-based methods as identified on a theoretical basis in section 2.2.1.

Apart from the diagnostics for civil aero-gas turbines, Pratt&Whitney have developed estimation techniques for military turbofan engines. This work has been carried out with Mc Donnell Aircraft Company and NASA and the applications include diagnostics, in-flight performance optimisation and adaptive control (Luppold et al., 1989; Kerr et al., 1992). In particular PW studies are closely related to NASA works about Performance Seeking Control (PSC) algorithms (Maine et al., 1990; Alag and Gilyard, 1990; España, 1993; España and Gilyard, 1993). In these applications emphasis is put on real-time algorithms and often the final aim is not the calculation of engine performance parameters. Nonetheless, many hints can be extracted which can be useful for gas turbine diagnostics in general.

The main features of the real-time diagnostic system developed by PW are the following:
- a full non-linear model of the engine (the so-called State Of the Art Propulsion Program, SOAPP) is linearised to get a piecewise linear State Variable Model (SVM) of the engine

- real-time implementation is pursued
- a common KF is used
- the state vector (made by LP and HP spool speed and composite metal temperature) is augmented by performance parameters
- a so-called *Wiener* dynamic model is chosen for the performance parameters:

$$\dot{\mathbf{x}} = \mathbf{w} \tag{2.42}$$

  where $\mathbf{w}$ is white Gaussian zero-mean noise.
  The justification for this choice is that this kind of process is an appropriate statistical representation for a slow-varying parameter and its implementation is easy too. The process noise covariance is chosen by tuning.
- tuning is used to desensitise the filter design to biases
- the proposed algorithm does not directly account for possible sensor biases, as the engine is supposed to be provided with a Full Authority Digital Engine Control system (FADEC). FADEC systems possess some level of redundancy in their flight critical sensors and are used along with Fault Detection and Isolation (FDI) algorithms
- accuracy and reliability of the KF are good, even in multiple fault conditions
- since the matrices describing the system are time invariant, the KF gain matrix converges to a unique steady state if the system is both controllable and observable (Gelb, 1974). This means that KF gain matrix can be calculated off-line by using numerical packages able to solve the Riccati equation
- whereas large sensor errors are assumed to be detectable by hardware redundancy and FDI systems, small sensor errors affect the diagnostic accuracy by changing the values of the performance parameters. Measurement biases that cannot be explained in terms of component deviations result in quasi-steady unresolved residuals. In an ideal case (perfect model and high observability), the residual magnitude would be equal to the bias magnitude (Kerr et al., 1991). Limitations imposed by the existing control sensors imply that fault isolation relying on this property of residuals is not possible, even though the estimation algorithm shows a good insensitivity to bias for most parameters. In conclusion, the current set of sensors is insufficient to separately identify the engine performance parameters and sensor biases. Analytical proof is given by España (1993) that the biases cannot be estimated unless more unbiased measures are made available. Reynolds effects and modelling errors cannot be distinguished from variations in performance parameters as well (Maine and Gilyard, 1990)
- a Luenberger observer (Gelb, 1974) can be used rather than a KF to simplify the implementation of the filter (España and Gilyard, 1993; España, 1993). The final diagnostic accuracy is not reduced
- observability studies can be made in order to optimise the choice of the instrumentation set necessary to detect a given set of performance parameters.

In conclusion, PW's and NASA's point of view is that no distinction is possible between sensor biases and engine component deterioration with the current instrumentation systems and estimation algorithms.

2.3 Multiple Operating Point Analysis

The research team from the University of Athens focused on some of the limitations affecting the common GPA and therefore developed the Multiple Operating Point Analysis (MOPA), a generalised form of GPA able to overcome the problem of the small number of sensors, measurement noise and in somehow sensor faults.
The influence of the operating condition on the relation between measurements and performance parameters is made explicit (Stamatis and Papailiou, 1988)

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \mathbf{w}) \tag{2.43}$$

Eq. (2.43) is basically the same as (2.1) but here the vector $\mathbf{w}$ of the environment and power setting is written. The standard GPA linearises the equation with respect to the performance parameter vector:

$$\Delta\mathbf{z} = ICM(\mathbf{w}) \cdot \Delta\mathbf{x} \tag{2.44}$$

MOPA enables to extract useful diagnostic information by applying eq. (2.44) at several operating conditions, that is for different $\mathbf{w}$'s. This should be useful due to the non-linear relationship between the ICM elements and the working point setting parameters (Stamatis et al., 1989).
Measurement noise is accounted for:

$$\Delta\mathbf{z}_i = ICM(\mathbf{w}_i) \cdot \Delta\mathbf{x} + \mathbf{v}_i \qquad i = 1,...,N_P \tag{2.45}$$

where N_P discrete operating points are considered and the measurement noise is zero-mean, white and Gaussian as usual. The dynamic equation associated with (2.45) is:

$$\Delta\mathbf{x}_{i+1} = \Delta\mathbf{x}_i \tag{2.46}$$

Eq. (2.46) means that the time intervals between two consequent reading are long enough to let the engine reach a steady state and short enough to consider the deterioration constant.
The standard linear least-squares technique is used to get a minimum variance unbiased estimate:

$$\Delta\mathbf{x} = P_N \sum_{i=1}^{N_P} ICM_i^T \cdot R_i \cdot ICM_i \cdot \Delta\mathbf{z}_i \tag{2.47}$$

where:

- R_i is the measurement noise covariance matrix
- P_N is the state error covariance matrix and can be calculated as follows:

$$P_N^{-1} = \sum_{i=1}^{N_P} ICM_i^T \cdot R_i^{-1} \cdot ICM_i \tag{2.48}$$

Eq. (2.48) represents the so-called GPA model of order N_P.
A Performance Estimation Uncertainty Index (PEUI) is introduced, which is a measure of the diagnosis accuracy:

$$J = \sqrt{\frac{1}{M} tr(P_N)} \tag{2.49}$$

where M is the number of measurements and $tr(P_N)$ is the trace of the matrix P_N.

The following remarks can be made regarding the MOPA:

- Diagnostics should be possible even with a small number of sensors.
- Better accuracy is claimed when the operating conditions are far from each other. The method's accuracy has actually been tested by the Greek team with simulated data. When deteriorated engine performance is simulated, as no exact information about the actual map's modification is usually available, the fault's effect is obtained by a simple shift of the map itself. If this simplification is accepted, utilisation of operating points located far from one another on the map should enable maximum exploitation of the model's non-linearity and then of the available instrumentation set.
  The technique's underlying assumption is obviously that changing operating condition does not modify the value of the performance parameter variation. On the other hand Doel (1994b) noted that a different working point means different aerodynamic conditions and in this sense efficiencies and flow capacities can significantly change with the operating condition. The deviation though can be supposed small provided the power settings are close to each other. That is why Doel suggests that only two operating conditions should be considered: take-off and maximum continuous power, as they are close aerodynamically.
  In conclusion, as the diagnostic method is to be used with real data, the operating points should be chosen close to one another.
- Increasing the number of working points betters the estimation, as the information acquired is enlarged
- The more non-linear is the functional relationship between the ICM elements and the environmental and power setting parameters the larger is the useful information available from multiple readings
- Measurement noise is accounted for
- Cross checking on the measured values enables a sort of sensor fault detection and isolation
- Observability has to be checked, because in theory fault diagnosis is claimed to be feasible even with few sensors
- Whereas the technique is suitable for test-cell diagnostics, it has no tracking capability and therefore it is not particularly suitable for on-wing diagnostics.

2.4 The German approach

Interesting studies have been made by the University of the German Armed Forces, Hamburg. Experimental data usually come from an RB199 turbofan engine. The basics of the German approach are the following:

- a linearised approach is pursued and the reference point for the Taylor series is a "fault-free" condition
- measurement noise and sensor errors are accounted for (mainly in terms of offsets and drifts)
- the characteristics of the engine components are assumed to be known only approximately and optimisation techniques are used to adapt the maps to real conditions
- test cell diagnostics is usually carried out with measurements coming from more than one operating condition
- unconventional estimation techniques, such as expert systems, fuzzy logic and neural networks, are used together with conventional model-based techniques
- the number of measurements is always larger than the number of performance parameters (hardware redundancy).

The model-based diagnostics proposed by Lunderstaedt (Lunderstaedt and Fiedler, 1988) relies on a linear GPA where the characteristic maps are assumed to be known only at the fault-free condition.
The basic vector equation (2.1) is now written as follows:

$$\mathbf{x} = \mathbf{f}(\mathbf{z}) \tag{2.50}$$

where it is reminded that $\mathbf{z}$ is the measurement vector, $\mathbf{x}$ is the performance parameter vector and $\mathbf{f}$ is a non-linear vector function. $\mathbf{f}$ of eq. (2.50) is related to $\mathbf{h}$ of (2.1) as follows:

$$\mathbf{f} = \mathbf{h}^{-1} \tag{2.51}$$

Eq. (2.50) is expanded in Taylor's series around an operating point $\mathbf{x}_0$ and evaluated in a fault-free condition $\mathbf{x}_{ff}$ and a faulty one $\mathbf{x}$ respectively:

$$\mathbf{x}_{ff} \cong \mathbf{x}_0 + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{z}}\right|_{0\_CHAR} (\mathbf{z}_{ff} - \mathbf{z}_0) \tag{2.52}$$

$$\mathbf{x} \cong \mathbf{x}_0 + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{z}}\right|_{0\_MEAS} (\mathbf{z} - \mathbf{z}_0) \tag{2.53}$$

When the Taylor's expansion is evaluated in the fault-free case (2.52) the derivatives are calculated as gradients of the characteristics, whereas in the faulty case (2.53) they are

directly determined by the performance parameters definitions, provided sufficient measurements are available.
Subtracting eq. (2.52) from eq. (2.53) produces:

$$\mathbf{x} - \mathbf{x}_{ff} \cong [\left.\frac{\partial \mathbf{f}}{\partial \mathbf{z}}\right|_{0\_MEAS} - \left.\frac{\partial \mathbf{f}}{\partial \mathbf{z}}\right|_{0\_CHAR}] \cdot (\mathbf{z} - \mathbf{z}_{ff}) \tag{2.54}$$

The calculation of the quantity in brackets requires the knowledge of inlet and outlet thermodynamic conditions of the flow and of the characteristics' gradients. The latters are usually known only approximately and therefore an estimation technique is utilised to calculate them.
After normalisation, eq. (2.54) is re-written as follows:

$$\Delta\mathbf{x} = Q \cdot \Delta\mathbf{z} \tag{2.55}$$

Simulation of different errors and the use of a least-square estimation make it possible to calculate the gradients and optimise the elements of the matrix Q. Details about the estimation are provided in appendix B. The application of the optimisation to calculate corrected values of the characteristics' gradients produces accurate diagnostic results.

After the optimisation, performance parameters and sensor errors are calculated according to classic estimation techniques. The ideal equation (2.55) is modified as follows:

$$\Delta\mathbf{x} = Q \cdot (\Delta\mathbf{z} - \delta\Delta\mathbf{z} - \mathbf{v}) \tag{2.56}$$

where $\delta\Delta\mathbf{z}$ is the systematic sensor error vector and $\mathbf{v}$ is the measurement noise vector.
The estimation can be carried out in two different ways: if such a large number of measurement samples are available that the applied filter produces precise estimates of the mean values, then the superposition principle can be used and noise filtering and systematic errors isolation can be carried out separately. Otherwise, the two tasks have to be worked out at the same time.
An outline of the two-step technique is reported below.
1) ***Noise is filtered out***
Eq. (2.56) reduces to:

$$\Delta\mathbf{z} = C \cdot \mathbf{x} + \mathbf{v} \tag{2.57}$$

where C is the pseudo-inverse of Q:

$$C = Q^T (QQ^T)^{-1} \tag{2.58}$$

Assuming the noise to be normally distributed, the common minimum variance, maximum likelihood estimation provides in the case of single measurement vector (snapshot calculation):

$$\Delta\hat{\mathbf{x}} = \Delta\mathbf{x}_0 + (P_0^{-1} + C^T R^{-1} C)^{-1} C^T R^{-1} (\Delta\mathbf{z} - C \cdot \Delta\mathbf{x}_0) \tag{2.59}$$

In case of no prior information on the state eq. (2.59) reduces to:

$$\Delta\hat{\mathbf{x}} = (C^T R^{-1} C)^{-1} C^T R^{-1} \cdot \Delta\mathbf{z} \tag{2.60}$$

In case a set of r measurement samples are available a non-recurrent version of the Bayesian estimator (2.59) is used:

$$\Delta\hat{\mathbf{x}} = (\frac{1}{r} P_0^{-1} + C^T R^{-1} C)^{-1} (\frac{1}{r} P_0^{-1} \Delta\mathbf{x}_0 + \frac{1}{r} C^T R^{-1} \sum_{i=1}^{r} \Delta\mathbf{z}_i) \tag{2.61}$$

In case of no prior information eq. (2.61) reduces to:

$$\Delta\hat{\mathbf{x}} = \frac{1}{r} (C^T R^{-1} C)^{-1} C^T R^{-1} \sum_{i=1}^{r} \Delta\mathbf{z}_i \tag{2.62}$$

It can be shown that in the limiting case of $r \to \infty$ (2.61) becomes (2.62). This means that a good prior knowledge can substantially reduce the number of measurements needed for an accurate diagnosis. The usual problems of tuning are claimed.

2) Systematic sensor errors are isolated and evaluated

Eq. (2.56) reduces to:

$$\Delta\mathbf{z} = C \cdot \Delta\mathbf{x} + \delta\Delta\mathbf{z} \tag{2.63}$$

If M is the number of sensors, N the number of performance parameters and M_{bias} the number of sensor offsets (with $M_{bias} \leq M$), an $(M_{bias}, 1)$ vector of offsets is introduced:

$$\delta\Delta\mathbf{z} = G \cdot \mathbf{s} \tag{2.64}$$

and eq. (2.63) becomes:

$$\Delta\mathbf{z} = C \cdot \Delta\mathbf{x} + G \cdot \mathbf{s} \tag{2.65}$$

If G is supposed to be known, the estimate is:

$$\Delta\hat{\mathbf{x}} = [(C^T - \Omega) \cdot C]^{-1} (C^T - \Omega) \cdot \Delta\mathbf{z} \tag{2.66}$$

$$\hat{\mathbf{s}} = (G^T G)^{-1} G^T \cdot (\Delta\mathbf{z} - C\Delta\mathbf{x}) \tag{2.67}$$

where:

$$\Omega = C^T G(G^T G)^{-1} G^T \tag{2.68}$$

The estimation provided by (2.66), (2.67) and (2.68) is basically a least-squares calculation.

If the C matrix has rank M, then $M_{bias} = M - N$ sensor errors are detectable. This number is usually too small because of the limited number of sensors present on board of jet engines. Therefore, several operating points of the gas turbine are considered in order to extract useful information and estimate the state of the sensors (see section 2.3 for details). A set of N_P measurement vectors are supposed to be available and the performance parameters (i.e. the state of the engine modules) are assumed to be the same at the different operating conditions. Under this assumption a technique is developed (Lunderstaedt, 1988) which makes the estimation of $M_{bias} \leq N_P \cdot M - N$ sensor errors possible. It relies on the application of the standard least-squares method to a series of equations similar to (2.65).

An underlying problem in the estimation technique outlined above is the choice of the matrix G. Provided a sufficient number of operating points is available (i.e. $N_P \geq 1 + \frac{N}{M}$), it can be assumed that many sensors are affected by errors and thereby the estimation can be carried out. However, the least-squares basis of the technique will cause a "smearing" effect. This can be avoided by combining the estimation technique with a hypothesis test to isolate the biased sensors, which will be included in the final estimation calculation.

It can be shown that the number of fault hypotheses to be taken into account in case of M_{bias} simultaneous sensor errors is:

$$h = \frac{M!}{M_{bias}!(M - M_{bias})!} \tag{2.69}$$

These h hypotheses are $H_{p,\ldots\ldots,q}$, where $p = 2,\ldots,M,\ldots\ldots; q = 1,\ldots\ldots,M + 1 - M_{bias}$.

The decision criterion relies on the fact that in a sensor error-free case the state is $\Delta\mathbf{x}$ in every operating condition:

$$\Delta\mathbf{z}_i = C_i \cdot \Delta\mathbf{x} \qquad i = 1,\ldots\ldots,N_P \tag{2.70}$$

If sensor errors are present, eq. (2.70) does not apply.

The trace of the (N,N) dimensional matrix is calculated for each hypothesis:

$$trace(R_{p,\ldots\ldots,q}) = \frac{1}{N_P - 1} \sum_{j=1}^{N} \sum_{i=1}^{N_P} [(\Delta x_{j,p,\ldots\ldots,q})_i - (\Delta x_{j,p,\ldots\ldots,q})_{mean}]^2 \tag{2.71}$$

where $(\Delta\mathbf{x}_{p,\ldots\ldots,q})_{mean}$ is the mean value over N_P.

The hypothesis $H_{p,\ldots\ldots,q}$ is to be accepted if

$$trace(R_{p,\ldots,q}) = 0 \tag{2.72}$$

Apart from sensor errors, the estimation will be affected by modelling errors, which prevent eq. (2.72) from applying. Therefore, the correct hypothesis will be provided by:

$$\min_{p,\ldots,q}(trace(R_{p,\ldots,q})) \tag{2.73}$$

The use of the estimation technique in conjunction with the hypothesis test seems to be very effective: time varying (steps and drifts) multiple sensor errors (even 3) are detectable.

Another model-based approach (Lunderstaedt, 1990) has been pursued which is based on eq. (2.56) rather than eq. (2.57) and (2.63). A least-squares technique using a set of measurements taken at different operating conditions has been developed, which provides a minimum variance unbiased recursive estimate. After noise is filtered out the model equation is:

$$(I + E_i)M_i \cdot \Delta\mathbf{x}_0 + Q_i N_i \Gamma \cdot \boldsymbol{\beta}_0 = Q_i \cdot \Delta\mathbf{z}_i \qquad i = 1,\ldots,N_P \tag{2.74}$$

where:

- Q_i is the matrix for the i-th working point
- M_{bias} out of M measurements are assumed to be affected by errors which are expressed as follows:

$$\mathbf{s} = \Gamma \cdot \boldsymbol{\beta} \tag{2.75}$$

  where $\mathbf{s}$ is the $(M,1)$ vector for measurement systematic errors, Γ is an (M,M_{bias}) matrix and $\boldsymbol{\beta}$ is a $(M_{bias},1)$ vector. Sensor errors related to different working points are expressed in terms of a single sensor error vector:

$$\mathbf{s}_i = N_i \cdot \mathbf{s}_0 \tag{2.76}$$

  and hence

$$\mathbf{s}_i = N_i \Gamma \cdot \beta_0 \tag{2.77}$$

- the state vector $\Delta\mathbf{x}_i$ for the i-th operating point is expressed in terms of a single condition through known matrices M_i :

$$\Delta\mathbf{x}_i = M_i \cdot \Delta\mathbf{x}_0 \tag{2.78}$$

- model errors are explicitly taken into account as $\delta\Delta\mathbf{x}_i$:

$$\delta\Delta\mathbf{x}_i = E_i M_i \cdot \Delta\mathbf{x}_0 \tag{2.79}$$

where E_i is a diagonal matrix with the non-zero elements representing the model errors being sought.

It should be noted that a more comprehensive approach can be developed, which estimates the errors of every element of the Q matrix. The computational burden though would be largely increased.

A cost function is introduced:

$$J = (1-\rho)\cdot f_1(\Delta\mathbf{x}_0, E_1, ..., E_p) + \rho \cdot f_2(\boldsymbol{\beta}_0) \tag{2.80}$$

where:

ρ is a constant to be chosen empirically and

$$f_1 = \sum_{i=1}^{p} \delta\Delta\mathbf{x}_i^T \cdot \delta\Delta\mathbf{x}_i \tag{2.81}$$

$$f_2 = \boldsymbol{\beta}_0^T \cdot \boldsymbol{\beta}_0 \tag{2.82}$$

The estimation of $\Delta\mathbf{x}_0$, $\boldsymbol{\beta}_0$ and $\delta\Delta\mathbf{x}_i$ is obtained iteratively. At first model errors are neglected and $\Delta\mathbf{x}_0$ and $\boldsymbol{\beta}_0$ are estimated from the following model (the same as (2.74) with no E_i):

$$M_i \cdot \Delta\mathbf{x}_0 + Q_i N_i \Gamma \cdot \boldsymbol{\beta}_0 = Q_i \cdot \Delta\mathbf{z}_i \qquad i = 1, ..., N_P \tag{2.83}$$

Then $\delta\Delta\mathbf{x}_i$ are calculated according to the following minimisation procedure:

$$\frac{\partial J}{\partial \boldsymbol{\varepsilon}_i} = 0 \qquad i = 1, ..., N_P \tag{2.84}$$

where:

$$\boldsymbol{\varepsilon}_i = E_i \mathbf{e} \tag{2.85}$$

with $\mathbf{e}$ as the unity vector. Eq. (2.83) can be adequately expanded and calculated.

The estimated model errors are now introduced in eq. (2.74) and the new couple of vectors $\Delta\mathbf{x}_0$ and $\boldsymbol{\beta}_0$ is used to calculate a new value for $\delta\Delta\mathbf{x}_i$ and so on.

It is worth pointing out that the underlying assumption of the above estimation is that both sensor errors and performance parameters are not dependent on the power setting. This means that engine and sensor fault vectors are supposed not to be changed by a shift in the power setting.

The assumption about the performance parameters is also the basis for the Multiple operating Point Analysis (MOPA) developed by Stamatis et al. (Stamatis and Papailiou, 1988; Stamatis et al., 1989). Again, see section 2.3 for a detailed discussion on the applicability of the technique.
Lunderstaedt (1988) has dealt with the problem of the effect on sensor errors of the operating condition in a greater detail. He uses the concept of sensor characteristic, that is a relation, usually non-linear, between the true value z_0 of the quantity and the measured quantity z_m:

$$z_0 = g(z_m) \tag{2.86}$$

where the function g is assumed quadratic:

$$z_0 = z_m + \Delta z \tag{2.87}$$

$$\Delta z = a + bz_m + cz_m^2 \tag{2.88}$$

If several sensors are present, eq. (2.88) becomes vectorial:

$$\Delta \mathbf{z} = \mathbf{a} + L \cdot \mathbf{b} + P \cdot \mathbf{c} \tag{2.89}$$

where the vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ have to be determined. If N_P working points are considered:

$$\Delta \mathbf{z}_i = N_i \cdot \mathbf{a}_1 + L_i \cdot \mathbf{b}_1 + M_i P_i \cdot \mathbf{c}_1 \qquad i = 1,....,N_P \tag{2.90}$$

Once again, a number M_{bias} less than M of sensors is assumed to be affected by errors.
In conclusion, this technique allows sensor errors to be taken into account by adding the three $(M_{bias},1)$ vectors to the state vectors to be estimated. Lunderstaedt claims that a quadratic function approximates the actual variation of sensor errors with power setting well, even though a basically linear relationship often comes out of the estimation.

In conclusion, some advantages of the model-based methods outlined above are the following:

- sensor faults are taken into account
- due to the use of hypotheses tests, an acceptable accuracy should be achievable (the smearing effect should be reduced). Two or three simultaneous sensor faults seem to be detectable accurately
- even though the characteristic maps may be known only approximately, the optimisation ensures a good diagnostic accuracy.

However, some disadvantages are:

- in case of large degradations, the linearised approach may impair the diagnostics
- a comprehensive instrumentation set is required

- if sensor characteristics are used, a thorough experimental validation of the assumptions should be made

The techniques described so far are strictly model-based. The measurement uncertainties, namely measurement noise, sensor errors and poor instrumentation set may also be tackled with analytical techniques which are broadly called Artificial Intelligence (AI) methods. Neural Networks (NNs), Knowledge-Based Systems (KBSs), Fuzzy Logic Systems (FLSs) and Genetic Algorithms (GAs) are widespread AI techniques which may have some advantages over the conventional model-based methods. Many researchers and the main manufactures (Doel, 1994b; Doel and LaPierre, 1989; Provost, 1995; Rolls-Royce Magazine, 1993) believe that further work should be carried out in this area to exploit the capabilities of AI techniques to cope with the typical uncertainties encountered in gas turbine diagnostics. Following this trend, the German research team has tried to combine model-based and AI techniques in many ways. A brief outline of the published work is given below.

The use of knowledge-based systems (Lunderstaedt and Hillemann, 1992) and fuzzy logic (Lunderstaedt and Hillemann, 1993) makes it possible to develop a diagnostic system that relies on a single working point.
The model upon which the diagnostics is based is given by:

$$\Delta \mathbf{z} = C \cdot \Delta \mathbf{x} \tag{2.91}$$

which stems from (2.56) by filtering out the measurement noise. It is noted here that due to problems related to the presence of the pseudo-inverse matrix of eq. (2.58) an alternative technique of calculating C is introduced (Lunderstaedt and Hillemann, 1992). Detection and isolation of faulty sensors is accomplished by means of a KBS. Since all states are normalised so as to be zero in the fault-free condition, states representing efficiencies are supposed to be non-negative. Thereby two kinds of rules are considered:

- state-based rules: the effects on one particular state of all sensors are analysed
- sensor-based rules: the effects on all states of a fault in a particular sensor are analysed

Fault trees are built based on the signs of the elements of the matrix Q. This means that only three conditions are allowed: positive, negative and zero. State-based and sensor-based rules provide two sets of possible faulty sensors respectively. The intersection of the two sets selects the sensors that are most likely to be faulty for the given input measurement vector. In this way the expert system is able to reduce from M to L the number of sensors to be analysed.
The subsequent diagnostics can be performed in two different ways, as described below.
1) ***Model-based minimisation.***
Combining (2.91) and (2.55) provides:

$$\Delta \mathbf{z}_A = CQ \cdot \Delta \mathbf{z}_E \tag{2.92}$$

where $\Delta \mathbf{z}_E$ is the input measurement vector and $\Delta \mathbf{z}_A$ is the output measurement vector. If the $\Delta \mathbf{z}_E$ is fault-free, then $\Delta \mathbf{z}_E$ is equal to $\Delta \mathbf{z}_A$.

If $\Delta \mathbf{z}_E$ is affected by faults, eq. (2.92) becomes:

$$\Delta \mathbf{z}_A = CQ \cdot (\Delta \mathbf{z}_E + \delta \Delta \mathbf{z}) \tag{2.93}$$

where $\delta \Delta \mathbf{z}$ is the measurement bias vector. In this case $\Delta \mathbf{z}_E$ is different from $\Delta \mathbf{z}_A$. This is exploited to calculate the measurement biases.
The canonic $(M,1)$ vector $\mathbf{e}_k$ is introduced, which has all elements equal to zero apart from the k-th element, which is equal to the unity:

$$\mathbf{e}_k = [00...1...000]^T \qquad k = 1,...,M \tag{2.94}$$

The corresponding output vector is:

$$\Delta \mathbf{z}_k = CQ \cdot \mathbf{e}_k \tag{2.95}$$

whereas for the real output vector eq. (2.92) applies.
A quadratic cost function is introduced:

$$S_{k1,2,...,N_P} = \sum_{j=1}^{M} [\Delta z_{Aj} - \sum_{i=1}^{N_P} \alpha_{kl} \Delta z_{klj}]^2 \tag{2.96}$$

and then minimised with respect to α_{kl}:

$$\min_{\alpha_{kl}} S_{k1,...,N_P} \tag{2.97}$$

In this way identification and quantification of the sensor errors are carried out simultaneously.
2) ***Correlation***
Eq. (2.92) is used with L input vectors of this kind:

$$\Delta \mathbf{z}_E = [0,....,0.01,...,0]^T \tag{2.98}$$

and hence L $\Delta \mathbf{z}_A$ are derived. The L vectors are those which have been selected by the expert system. A subsequent selection is made under the assumption that the best similarity between a real measurement vector $\Delta \mathbf{z}$ and one of the L stored patterns exists if the sensor faults affecting $\Delta \mathbf{z}$ are the same as those in $\Delta \mathbf{z}_A$. The superposition principle can be exploited because of the model linearity. The correlation factor to be maximised is:

$$C_k = \frac{1}{M} \sum_{j=1}^{M} e^{-V|\Delta z_{Akj} - \Delta z_j|} \tag{2.99}$$

When sensor errors are isolated, a common least-squares estimation technique can be used.

A more refined and perhaps more effective sensor error detection and isolation technique has been introduced by Lunderstaedt and Hillemann (1993), which makes use of fuzzy logic. In this case the states are defined in terms of six linguistic variables (positive big, positive medium, positive small/zero, negative small/zero, negative medium, negative big). Every variable has its own piecewise linear membership function. Sensor errors are detected on the basis of their typical features, even in case of multiple faults. The maximum number of fault affected measurements is assumed equal to 3. For every fault or fault combination noise is filtered out, eq. (2.55) is used and a *pattern value* is calculated:

$$P = \frac{1}{M_{feat}} \sum_{i=1}^{M_{feat}} \mu_i(x_j) \qquad j = 1,...,N \qquad (2.100)$$

where M_{feat} is the number of features of the fault pattern and N is the number of the states. The higher is the pattern value, the higher is the likelihood that the considered fault(s) is(are) present. The sensor fault condition with the largest P is selected and the usual model-based minimisation method applied.
The claimed accuracy and reliability of the diagnosis is good.
The expert system developed (named XSD) shows some advantages over the model-based diagnostic system made by the same research team:

- it is robust
- it produces diagnostic answers by using a single operating condition.

Junk and Lunderstaedt (1995) used neural network techniques for gas turbine diagnostics. Sensor fault detection, isolation and accommodation are performed by two sets of neural networks and after the accommodation the model optimisation can be carried out. The various diagnostic steps are outlined below.
1) ***State vector generation***
The normalised version of eq. (2.55)

$$\Delta \mathbf{x} = QN_y^{-1} \cdot (\mathbf{z} - \mathbf{z}_0) \qquad (2.101)$$

where $N_y = diag(z_{01}, z_{02}, ..., z_{0M})$ is used with a faulty measurement vector to produce the relative state vector:

$$\Delta \mathbf{x}_F = QN_y^{-1} \cdot (\mathbf{z} - \mathbf{z}_0 + \Delta \mathbf{z}_F) \qquad (2.102)$$

2) ***Reconstruction of*** $\Delta \mathbf{z}_F$
Here M two-layer NNs calculate $\Delta \mathbf{z}_F$ from the input $\Delta \mathbf{x}_F$. Each net calculates only one component of the vector. Learning of the nets is made by using a series of

measurement vectors $\mathbf{z}_k$, $k = 1,2,...$ and superimposing randomly generated sensor fault vectors.

3) ***Classification***

A NN distinguishes between sensor fault and engine module fault. Its inputs are both $\Delta\mathbf{z}_F$ and $\Delta\mathbf{x}_F$ and the output permits the distinction.

4) ***Model optimisation***

After a certain operating time of the gas turbine new readings are taken and used to adapt the engine model. The vector to be determined is:

$$\mathbf{t} = [\rho_1, \varepsilon_1, ..., \rho_n, \varepsilon_n, z_{01}, z_{02}, z_{03}]^T \qquad (2.103)$$

where ρ_i and ε_i are the same gradients of the model-based technique detailed in appendix B and z_{0i} are the measurements defining the working point (e.g. ambient temperature and pressure, fuel flow).

The cost function to be minimised is the following:

$$S(\mathbf{t}) = \sum_{k=1}^{M_{meas}} (\Delta\mathbf{x}_k^T \cdot \Delta\mathbf{x}_k) + \mathbf{K}_p^T \cdot \mathbf{p}(\mathbf{t}) + \mathbf{K}_S^T \cdot \mathbf{m}(\mathbf{t}) \qquad (2.104)$$

where:

- M_{meas} is the number of measurement vectors used for the adaptation
- the second term is a weighted quadratic form of the penalty function which prevents the characteristics' gradients to overflow
- the third term accounts for the standard deviations of the state vector components due to the measurement noise

Minimisation of the cost function is performed by Levenberg-Marquardt algorithm.

Even though NN suitability for diagnostics will be discussed in chapter 3 in a greater detail, the robustness to measurement noise and the capability of working without a model have to be highlighted. As far as the simulation results are concerned, the combined model-based-NN diagnostic system outlined above was able to detect and isolate single sensor faults. The accommodation was achieved in case of offset but a bias in the estimation remained in case of slight drift.

2.5 Robust diagnostics using eigenstructure assignment

A novel fault detection scheme for jet engines has recently been developed by Patton (Patton and Chen, 1991). It relies on a technique named *eigenstructure assignment*, applied to an estimator similar to the Kalman Filter (Luenberger observer, see Gelb, 1974). The main aim of the work is the development of a diagnostic technique insensitive to disturbances whilst being highly sensitive to sensor faults.

The model for which the estimation technique is developed is made of the measurement and the dynamic model equations:

$$\dot{\mathbf{x}}(t) = A \cdot \mathbf{x}(t) + B \cdot \mathbf{w}(t) + E \cdot \mathbf{d}(t) \tag{2.105}$$

$$\mathbf{z}(t) = C \cdot \mathbf{x}(t) + D \cdot \mathbf{w}(t) + \mathbf{f}(t) \tag{2.106}$$

where:

- $\mathbf{x}(t) \in R^N$ is the state vector
- $\mathbf{w}(t) \in R^P$ is the known input vector
- $\mathbf{d}(t) \in R^S$ is the unknown input disturbance vector acting on the system dynamics
- $\mathbf{z}(t) \in R^M$ is the measurement vector
- $\mathbf{f}(t) \in R^M$ is the sensor fault vector
- A, B, C, D, E are matrices assumed to be known.

In the case of gas turbine fault detection, C is the identity matrix, D is the zero matrix and E can be estimated as shown later on.
A Luenberger observer is used to estimate the state vector:

$$\dot{\hat{\mathbf{x}}}(t) = (A - KC) \cdot \hat{\mathbf{x}}(t) + (B - KD) \cdot \mathbf{w}(t) + K \cdot \mathbf{z}(t) \tag{2.107}$$

$$\hat{\mathbf{z}}(t) = C \cdot \hat{\mathbf{x}}(t) + D \cdot \mathbf{w}(t) \tag{2.108}$$

The state estimation error is defined as follows:

$$\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t) \tag{2.109}$$

The state estimation dynamics results:

$$\dot{\mathbf{e}}(t) = A_c \cdot \mathbf{e}(t) + E \cdot \mathbf{d}(t) - K \cdot \mathbf{f}(t) \tag{2.110}$$

where $A_c = A - KC$.
A (p,1) residual vector is defined as:

$$\mathbf{r}(t) = W \cdot \mathbf{e}_z(t) = W \cdot (\mathbf{z}(t) - \hat{\mathbf{z}}(t)) \tag{2.111}$$

where W is a (p,m) weighting matrix.
Using the above definitions provides:

$$\mathbf{r}(t) = WC \cdot \mathbf{e}(t) + W \cdot \mathbf{f}(t) = H \cdot \mathbf{e}(t) + W \cdot \mathbf{f}(t) \tag{2.112}$$

where $H = WC$ is a (p,n) matrix.
Since (2.110) and (2.112) apply simultaneously, the residuals result non-zero even if no fault affects the instrumentation set. This means that it can be difficult to distinguish the effects of the faults from the effects of the disturbances acting on the system.
The idea is to choose W and K so as to ***decouple residuals from disturbances***.

Turning to the Laplace transform space, the transfer function representing the effect of disturbances on residuals is:

$$G_{rd}(s) = WC[sI - (A - KC)]^{-1}E \tag{2.113}$$

In order to decouple the residuals from the disturbances the following condition has to be imposed:

$$WC[sI - (A - KC)]^{-1}E = 0 \tag{2.114}$$

The matrices W and K have to be chosen according to eq. (2.114). This is done through assignment of the eigenstructure of the system.
It can be shown (Patton and Chen, 1991) that if $WCE = 0$ and all rows of the matrix H are the left eigenvectors of A_c, the condition (2.114) is satisfied and the residuals are completely decoupled from the disturbances.

If the required eigenstructure is assignable, then perfect decoupling is achievable, otherwise the eigenvectors must be chosen close to the desired eigenvectors in a least square sense. In this case, the residuals will show low sensitivity to uncertainties and disturbances due to approximate decoupling.
An underlying problem of the technique outlined above is that the matrix E, deciding which sensors are affected by faults, is not known. Patton and Chen propose an estimation technique able to evaluate the distribution matrix E. The method assumes that the system disturbances are slowly varying and thereby an observer augmented with the disturbance vector is introduced. All the elements of the disturbance vector are supposed to have the same time dependence and an optimisation is carried out by using a Singular Value Decomposition method. The involved calculations are quite straightforward.

The eigenstructure assignment technique has been applied to data coming from a full non-linear simulation model of a two-spool turbofan engine. The measurement and dynamic system equations (i.e. eq. (2.105) and (2.106)) are obtained by linearising the non-linear model and by further reducing the obtained linear model made of 17 states to get a 5 state linear model. This may be done by the Schur model reduction method.

The following remarks on the eigenstructure assignment technique can be made:

- the unavoidable uncertainties present in the modelling are explicitly taken into account and accommodated. This robust approach is therefore much better than the KF for detecting sensor faults and the amount of prior knowledge required is about the same. It is worth reminding here that KF is not robust at all with respect to model inaccuracies and therefore leads to biased residuals. This undesired characteristic of the KF is highlighted even in a comparison study between KF and Neural Network techniques for Sensor Fault Detection, Identification and Accommodation (Napolitano et al., 1996)

- the robustness entails good prediction capability even though the non-linear behaviour of the engine is approximated by a linear system. The typical drawbacks due to linearisation of an actually non-linear system are very attenuated here
- the Schur reduction technique has two positive effects:
  1) it reduces the order of the system and therefore the computational burden
  2) it makes the disturbance direction almost constant, regardless of both the kind of input (a step and a sinusoidal input function have been used) and the operating point (70% and 95% of the HP spool speed have been used)
- the results obtained with the reduced 5 order system are very promising: even slowly varying sensor faults are quickly identified (the response is given with a minimal time transient), as the actual decoupling is achieved
- the technique can be expanded to estimate the status of the engine components through evaluation of the performance parameter vector
- the technique is suitable for analysis of the transient performance. However even a poorly dynamic modelling could be used for the estimation because of the robustness achievable
- both slowly and abruptly varying faults are detectable and noise can simply be considered as a disturbance on the measurements (Patton et al., 1986). The technique should be effective even with the high level of noise typical of gas turbine engine sensors.

2.6 Bayesian inference applied to gas turbine diagnostics

The Kalman Filter approach can be embedded in a more general framework, where estimation is regarded as a problem of making decisions under uncertainty. This is the Bayesian decision theoretic approach (Bryson and Ho, 1975).
Some attempts to apply Bayesian inference to GPA have been made and a brief outline of the investigations is shown.
According to Bryson and Ho, the following information is assumed to be given:
a) ***the measurement equation:***

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{v} \tag{2.115}$$

where:

- $\mathbf{z} \in R^M$ is the measurement vector
- $\mathbf{x} \in R^N$ is the performance parameter vector
- $\mathbf{v} \in R^M$ is the measurement noise, usually assumed to be Gaussian
- $\mathbf{h}$ is a vector function, possibly non-linear.

This information can be supposed to be available for gas turbines.
b) ***the joint probability density function:***

$$p(\mathbf{x}, \mathbf{v}) \tag{2.116}$$

Since measurement noise and performance parameters are assumed independent:

$$p(\mathbf{x},\mathbf{v}) = p(\mathbf{x}) \cdot p(\mathbf{v}) \tag{2.117}$$

$p(\mathbf{v})$ is calculated as follows:

$$p(\mathbf{v}) = p(v_1) \cdot p(v_2) \cdot \ldots\ldots \cdot p(v_M) \tag{2.118}$$

because the various measurement noises are independent. Usually $p(v_i)$ are available and are zero mean Gaussian.
The probability density function (pdf) for the state vector $\mathbf{x}$ is more difficult to get, because knowledge of the statistics of the engine components' deterioration is necessary. Nonetheless, this input is required even by the common KF and may however be modified through tuning.
Given these data, the Bayesian solution proceeds through four steps:
a) ***evaluation of*** $p(\mathbf{z})$
This could be done through Monte Carlo methods, which would involve long computing times.
An analytical approach is developed that relies only on eq. (2.115) and (2.116).
The measurement vector is augmented by the measurement noise, so that the following new state vector is obtained:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z} \\ \mathbf{w} \end{bmatrix} \tag{2.119}$$

The state vector is augmented by the measurement noise vector as well:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} \tag{2.120}$$

Eq. (2.115) becomes in augmented form:

$$\mathbf{Z} = \mathbf{F}(\mathbf{X}) \tag{2.121}$$

where:

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} \mathbf{F}_1(\mathbf{X}) \\ \mathbf{F}_2(\mathbf{X}) \end{bmatrix} \tag{2.122}$$

and:

$$\mathbf{F}_1(\mathbf{X}) = \mathbf{h}(\mathbf{x}) + \mathbf{v} = \mathbf{z} \tag{2.123}$$

$$\mathbf{F}_2(\mathbf{X}) = \mathbf{v} = \mathbf{w} \tag{2.124}$$

According to Jazwinski (1970), as an equation in the form of (2.121) has been reached, the following relation applies, which performs the calculation of $p(\mathbf{z})$:

$$p_{\mathbf{Z}}(\mathbf{Z}) = p_{\mathbf{X}}(\mathbf{F}^{-1}(\mathbf{Z})) \cdot \left| \frac{\partial \mathbf{F}^{-1}(\mathbf{Z})}{\partial \mathbf{Z}} \right| \tag{2.125}$$

where $\left| \frac{\partial \mathbf{F}^{-1}(\mathbf{Z})}{\partial \mathbf{Z}} \right|$ is the determinant of the Jacobian matrix indicated.

The right hand side of eq. (2.125) can be calculated, as detailed in appendix C. This enables the pdf $p_{\mathbf{Z}}(\mathbf{Z})$ and then $p(\mathbf{z})$ to be calculated, even if $\mathbf{h}$ is non-linear. In this case, as shown in appendix C, the most cumbersome calculation requires the evaluation of

$$\frac{\partial h_i^{-1}}{\partial z_j} \tag{2.126}$$

for all i and j. This can be performed by finite difference techniques and should not involve large memory and computing requirements.

b) ***evaluation of*** $p(\mathbf{x},\mathbf{z})$

According to Bryson and Ho (1975):

$$p(\mathbf{x},\mathbf{z}) = p(\mathbf{x},\mathbf{v}) \sqrt{\left| JJ^T \right|} \tag{2.127}$$

where

$$J = \frac{\partial [\mathbf{z} - \mathbf{h}(\mathbf{x})]}{\partial \mathbf{x}} \tag{2.128}$$

$|*|$ denotes the determinant of the matrix.

In the case of the gas turbine model, eq. (2.127) becomes:

$$p(\mathbf{x},\mathbf{z}) = p(\mathbf{x}) \cdot p(\mathbf{v}) \tag{2.129}$$

c) ***evaluation of*** $p(\mathbf{x} \mid \mathbf{z})$.

According to Bayes' theorem:

$$p(\mathbf{x}\,|\,\mathbf{z}) = \frac{p(\mathbf{x})\cdot p(\mathbf{v})}{p(\mathbf{z})} \tag{2.130}$$

It is noted that the knowledge of $p(\mathbf{x}\,|\,\mathbf{z})$, namely the *conditional probability density function* or *posterior density function,* is basically the solution of the estimation problem. This is due to the meaning of $p(\mathbf{x}\,|\,\mathbf{z})$: it is the pdf of $\mathbf{x}$ given a realisation of the measurement vector $\mathbf{z}$. Once $p(\mathbf{x}\,|\,\mathbf{z})$ has been evaluated, various criteria can be chosen to produce the estimation $\hat{\mathbf{x}}$ of $\mathbf{x}$:

- maximisation of the probability that $\mathbf{x} = \hat{\mathbf{x}}$. This means that $\hat{\mathbf{x}}$ is the value for which $p(\mathbf{x}\,|\,\mathbf{z})$ is maximum. In this case the common *maximum likelihood estimate* (ML) is obtained.
- minimisation of $\int |\mathbf{x} - \hat{\mathbf{x}}|^2 p(\mathbf{x}\,|\,\mathbf{z}) d\mathbf{x}$. The solution is the conditional mean estimate:

$$E(\mathbf{x}\,|\,\mathbf{z}) \tag{2.131}$$

This is the common *minimum variance estimate* (MV).

- minimisation of the maximum of $|\mathbf{x} - \hat{\mathbf{x}}|$. This is called *minimum error estimate.*

Due to the non-linearity of the problem, the three estimations do not usually produce the same results.

The estimation technique explained above is to be used for an assessment of the engine's health when a single measurement vector is available for diagnostics. A generalisation to multistage estimation can be developed if the following equation, describing the temporal evolution of the deterioration, is available beside eq. (2.115):

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{w}_{k+1}, t_k) \qquad k = 1,2,.... \tag{2.132}$$

where a discrete system is considered and $\mathbf{w}_k$ is the process noise at time t_k.

The key idea of the Bayesian approach is that of using the measurement to update the knowledge of the state of the system. In the multistage case, the updating procedure can be repeated every time a measurement is made. The posterior density function from the previous stage becomes the prior density function at the present stage. If $\mathbf{Y}_k$ is the set of measure vectors up to t_k :

$$\mathbf{Y}_k = [\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k]^T \tag{2.133}$$

the posterior density function after the measurement a time t_{k+1} is given by (Bryson and Ho, 1975):

$$p(\mathbf{x}_{k+1}\,|\,\mathbf{Y}_{k+1}) = \frac{\int p(\mathbf{z}_{k+1}\,|\,\mathbf{x}_{k+1})\cdot p(\mathbf{x}_{k+1}\,|\,\mathbf{x}_k)\cdot p(\mathbf{x}_k\,|\,\mathbf{Y}_k)\cdot d\mathbf{x}_k}{\iint p(\mathbf{z}_{k+1}\,|\,\mathbf{x}_{k+1})\cdot p(\mathbf{x}_{k+1}\,|\,\mathbf{x}_k)\cdot p(\mathbf{x}_k\,|\,\mathbf{Y}_k)\cdot d\mathbf{x}_k d\mathbf{x}_{k+1}} \tag{2.134}$$

The extrapolation of the pdf before the measurement is:

$$p(\mathbf{x}_{k+1}) = \int p(\mathbf{x}_{k+1} \mid \mathbf{x}_k) \cdot p(\mathbf{x}_k) d\mathbf{x}_k \tag{2.135}$$

Two quantities have to be calculated to use (2.134) and (2.135): $p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)$ and $p(\mathbf{z}_{k+1} \mid \mathbf{x}_{k+1})$. The way it can be done is shown below.

a) *Calculation of* $p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)$.

According to Jazwinski, an equation similar to (2.125) can be derived. In particular, as (2.132) can be written like this:

$$\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k, t_k) + \mathbf{w}_{k+1} \tag{2.136}$$

it can be shown that:

$$p(\mathbf{x}_{k+1} \mid \mathbf{x}_k) = p_{\mathbf{w}_{k+1}}(\mathbf{x}_{k+1} - \mathbf{g}(\mathbf{x}_k, t_k)) \tag{2.137}$$

where $p_{\mathbf{w}k+1}$ is the pdf of the process noise and as such assumed known.

b) *Calculation of* $p(\mathbf{z}_{k+1} \mid \mathbf{x}_{k+1})$.

According to Bayes' theorem:

$$p(\mathbf{z}_{k+1} \mid \mathbf{x}_{k+1}) = \frac{p(\mathbf{x}_{k+1}, \mathbf{z}_{k+1})}{p(\mathbf{x}_{k+1})} \tag{2.138}$$

In this case:

$$p(\mathbf{x}_{k+1}, \mathbf{z}_{k+1}) = p(\mathbf{x}_{k+1}) \cdot p(\mathbf{v}_{k+1}) \tag{2.139}$$

and hence

$$p(\mathbf{z}_{k+1} \mid \mathbf{x}_{k+1}) = p_{\mathbf{v}}(\mathbf{z}_{k+1} - \mathbf{h}(\mathbf{x}_{k+1})) \tag{2.140}$$

An assessment of the proposed single stage and multistage Bayesian estimation algorithm for gas turbine diagnostics is based on the following remarks:

- as the non-linearity is retained, the computational burden (computing time and memory) is necessarily large. The recursivity of the technique is not helpful in this case and moreover problems of convergence are likely to occur
- prior knowledge in the form of pdf for the performance parameters is required. This would affect the diagnostic result especially in the single stage case
- the usual problems of tuning of the KF are present here; the tuning is made more difficult and lengthy by the numerical complexity of the estimation
- a proper multistage estimation can be performed only if a model of the deterioration is available in the form of (2.132) or (2.136). This seriously limits the diagnostic capability of the system. Two cases are to be considered:

a) if a steady state diagnosis is pursued there exists no comprehensive equation of this kind
b) if a transient state diagnosis is chosen the dynamic equation is available, but the performance parameter vectors are likely to be constant during the transient and therefore the technique should be modified

- if a preliminary sensor and engine component fault detection and isolation system was used (relying for example on AI techniques), the estimation should be able to evaluate both performance parameters and sensor errors precisely due to the use of a fully non-linear (accurate) engine model.

In the light of the above points, a proper application of a Bayesian estimator should be embedded in a transient fault diagnosis system in which the dynamic equations are available and the performance parameters are assumed to be constant. This has been done by Katafygiotis (1991), who developed an approximate Bayesian estimator suitable for linear models in structural dynamical systems. A straightforward application of Katafygiotis' work to gas turbine fault diagnosis has been carried out by Consumi and D'Agostino (1997; 1998). A quite detailed review of the equations used in the Bayesian approach is contained in appendix D.

The following features characterise the Bayesian estimator developed:

- a dynamic model of the engine is used, where measured and unmeasured quantities are calculated as functions of the performance parameters and their standard deviations
- the model is linear, even though the technique should be equally applicable to non-linear systems
- the coefficients defining the performance parameter's shift in the maps are assumed to be constant throughout the sampling time
- the core of the estimation technique is Bayes' theorem, where the new probability density function of the performance parameters is updated according to an equation similar to (2.134)
- the technique is basically a Maximum Likelihood estimation, as a quadratic cost function is minimised to reduce modelling and measurement errors
- no problem of tuning of the estimation technique is encountered, because it can be shown that if the number of available readings is large, then a simple a priori probability density function, which is locally constant, can be used and this remarkably simplifies the method
- measurement non-repeatability noise and biases are considered
- the input quantities (fuel flow, flight speed, ambient pressure and temperature) are varied according to a sinusoidal function

As far as the performance of the estimator is concerned, the following important remark has to be done:

- the estimator is able to calculate the performance parameters even with a very small number of measurements (3 measurements for 6 performance parameters).

However, the following remarks are necessary to comprehensively evaluate the performance of the estimator when applied to gas turbine fault diagnosis:

- the validation of the model is done (Consumi and D'Agostino, 1997; 1998) by means of data coming from a simulation program using the same linear model used by the estimator. If no data coming from a real engine are available, a correct approach for the assessment of an estimation technique for gas turbine diagnostics requires the estimator's results to be compared with data produced by a fully non-linear model of the engine. The highly non-linear model of the engine may be linearised only if the linearisation ensures a good overall approximation of the actual behaviour. A problem of the same kind is encountered when linear Kalman filtering techniques are applied to highly non-linear processes
- if the estimation were applied to a practical case, the main sources of error would be given by:
  1) the "smearing" effect, as the basis of the technique is a ML estimation. This means that the actual diagnostic accuracy has to be ascertained when only a small number of performance parameters are affected by faults
  2) the ML nature of the estimation: if the non-linearity of the studied system is large, then the probability density function is such that its peak is far from the minimum variance solution. The more non-linear is the system, the less useful is the ML estimation. If the system is linear, the solution is the common MV estimation given by a Kalman filter
- for the technique to apply, a very large number of samples have to be available, otherwise the approximation of the prior probability density function with a constant value is not acceptable
- different measurement non-repeatability ranges have not so far been taken into account, as no weighting is provided among the various terms of the quadratic cost function to minimise. This means that the estimator is likely to produce inaccurate results when applied to actual engine data, because some measurements (i.e. spool speeds) are more accurate than others (mass flow) and this has to be properly accounted for. In appendix D a modification to the technique is given, which allows to account for different non-repeatabilities. However, to date no simulation has been carried out to ascertain the performance of the modified estimator.
- no problem of transients in the measurements is assumed
- to date, the technique seems to have a somewhat limited capability to cope with measurement biases and noise. Further simulations are necessary to assess the effects of measurement uncertainty on diagnostic accuracy
- the ratio of the final error to the imposed deviation of the performance parameters is usually quite high.

In conclusion, the Bayesian-ML estimator outlined above shows interesting potential but should be modified as suggested and thoroughly tested to ascertain its effectiveness to perform actual gas turbine fault diagnosis.

2.7 Non-linear Kalman filtering techniques applied to gas turbine diagnostics

Some of the drawbacks of Kalman filtering have been already introduced in section 2.2.1 and are listed here for convenience:
1. tuning
2. "smearing" effect
3. knowledge of the fault dynamics
4. non-linearity

Whereas the problems related to points 1, 2 and 3 affect the estimation performance in about the same way for both a linear and a non-linear process, the issue about non-linearity has to be further analysed. It is worth pointing out that when the system is characterised by non-linearity, the performance of the possible estimation techniques should be tested through simulation, as non-linear estimators' behaviour is somewhat unpredictable and the same applies to linear estimators used for non-linear systems. Nonetheless, some hints about the estimation performance can be extracted by the classic filtering theory. An analysis of this kind is attempted here.
For the moment complete knowledge of the fault dynamics, even in terms of process noise, is assumed.
Various possible situations are considered in the sequel.

a) If an actually linear system has to be processed, then the most sensible choice is a common Kalman filter, as it is optimal with respect to any reasonable minimisation criterion (especially minimum variance and maximum likelihood). Moreover the solution is achieved through a recursive technique, which can be very useful due to the limited computing power required.

b) If the system is actually non-linear, a linearisation can be done and then the common linear KF can be applied in a straightforward way. In this case, though, the difference between the actual and the simulated behaviour of the system may be large and may lead to divergence, i.e. ever increasing distance between the state and the estimate. In practice, the onset of divergence manifests itself by inconsistency of the residuals with their predicted statistics. Residuals become biased and larger as more measurements are collected and processed. A similar behaviour of the estimator is observed when a measurement becomes biased (Kerr et al., 1991). If divergence effects are added to the "smearing" effect typical of the KF, the estimation accuracy may become unacceptable. As far as gas turbine diagnostics is concerned the linearisation of a process characterised by such a large non-linearity as a gas turbine engine is probably responsible for inaccuracies of the estimation, especially when time varying multiple faults are present. In some instances, moreover, divergence does occur (Urban and Volponi, 1992).

c) If the effect of the non-linearity of the system on the estimation accuracy is ascertained, a non-linear version of the KF can be used to try to approximate the engine behaviour better. The most common and used filtering techniques are the Extended Kalman Filter (EKF) and the Iterated Extended Kalman Filter (IEKF). A detailed description of EKF and IEKF equations is given in appendix E. A brief comparison between the linear and the non-linear versions of the KF is given below.

It can be shown (Bryson and Ho, 1975) that the KF minimises the cost function given by (2.21), namely:

$$J=\frac{1}{2}(\mathbf{x}(0)-\hat{\mathbf{x}}_0)^T P_0^{-1}(\mathbf{x}(0)-\mathbf{x}_0)+\frac{1}{2}\sum_{i=1}^{k-1}\mathbf{w}_i^T Q_i \mathbf{w}_i+\frac{1}{2}\sum_{i=1}^{k}(\mathbf{z}_i-H_i\mathbf{x}_i)^T R_i^{-1}(\mathbf{z}_i-H_i\mathbf{x}_i) \tag{2.141}$$

Moreover, minimisation is achieved in a recursive fashion. For a linear problem, the minimum variance and the maximum likelihood coincide and therefore the KF can be considered the best choice, provided the modelling of the process is sufficiently accurate. If the process is non-linear, the cost function to minimise becomes:

$$J=\frac{1}{2}(\mathbf{x}(0)-\hat{\mathbf{x}}_0)^T P_0^{-1}(\mathbf{x}(0)-\hat{\mathbf{x}}_0)+\frac{1}{2}\sum_{i=1}^{k-1}\mathbf{w}_i^T Q_i \mathbf{w}_i+\frac{1}{2}\sum_{i=1}^{k}(\mathbf{z}_i-\mathbf{h}_i(\mathbf{x}_i))^T R_i^{-1}(\mathbf{z}_i-\mathbf{h}_i(\mathbf{x}_i)) \tag{2.142}$$

A solution minimising the cost function (2.142) ensures maximum likelihood and approaches minimum variance asymptotically as the number of measurements increases. Therefore, aim of a non-linear filter would be the minimisation of the cost function (2.142), possibly in a recursive way. It can be shown that the EKF and the IEKF produce biased and suboptimal estimates due to linearisation of the cost function. From a practical point of view, this means low accuracy in the estimation of the engine's health.

As a matter of fact, most non-linear least squares estimation algorithms require a choice between an optimal solution and a recursive formulation. If recursivity is a paramount requirement, then optimality is neglected. A third possible solution has actually been suggested by Haupt et al. (1995). The proposed estimation technique splits the problem of cost function minimisation into a linear first step and a non-linear second step by defining new first step states that are non-linear combinations of the unknown states. Estimates of the first step states are obtained by minimising the first step cost function with a KF, whereas estimates of the unknown, or second step states, are obtained by minimising the second step cost function with an iterative Gauss-Newton algorithm. The two step estimator is shown to be optimal for static problems in which the time variation of the measurement equation can be separated from the unknowns. From a practical point of view, Haupt showed that a better estimation accuracy is usually achievable with the two step filter than with an IEKF, especially when most of the information is found in the measurements and very little a priori information is available. More details about Haupt's estimator are given in appendix F.

As far as gas turbine diagnostics is concerned, it should be reminded that the fault dynamics, i.e. the process dynamics, is unknown and should somehow be estimated. Actually, the following choices are possible in general:

1. the fault dynamics is neglected, as the dynamic equation is simply given by:

$$\dot{\mathbf{x}}=0 \tag{2.143}$$

GE's approach uses this simple equation, along with a simplified version of the KF, namely an exponential smoother (Doel, 1994a). In this case no tuning is necessary, but the effect of such a coarse approximation on a non-linear estimation algorithm's performance should be tested.

2. the dynamics is approximated by assuming that the fault temporal evolution is slow and therefore a Wiener model is used:

$$\dot{\mathbf{x}} = \mathbf{w} \tag{2.144}$$

where $\mathbf{w}$ is white Gaussian noise whose covariance matrix has to be chosen through tuning. PW's real time KF uses this approximation and tuning is utilised to adjust the estimator's performance. The difference between the actual and the assumed dynamics is likely to have a large effect on the non-linear estimation algorithm's performance, which should be thoroughly tested.

3. the whole dynamics is estimated in some way. This is Provost's approach and the same remarks as in point 2 can be done.
4. Approximate dynamic models are used for various classes of engines (Urban and Volponi, 1992).

Actually, GE claims that fault temporal evolution models are going to be used to verify if improvements are viable (Doel, 1994b), but nothing more has been published on this subject. PW have shown that application of an approximated fault dynamics model introduces minor improvements in terms of accuracy.

It is worth pointing out that the problem underlying most of the pitfalls of KF is its inherently ***low robustness to unmodelled effects***: if measurement and dynamic models are a good approximation of the real system, then the performance of the estimator is good, but in the real case the amount of knowledge is very poor. As far as gas turbine diagnostics is concerned, this is due to both the unavailability of fault dynamics models and the frequent choice of a linearised model for the measurement equation, because of computational convenience. In an imprecise but effective way, it could be said that the Kalman filter is a "strongly" model based technique, especially when compared to other estimators such as the observer with eigenstructure assignment (section 2.5) or Artificial Intelligence techniques.

In conclusion, three choices are possible for gas turbine diagnostics in case the non-linearity of the problem is required to be taken into account:

1. application of IEKF. The actual performance of the non-linear version should be tested and properly analysed, as suboptimality problems are likely to occur, especially because of incorrect dynamics modelling.
2. minimisation of the non-linear cost function (2.142) through non-linear smoothing techniques. Various iterative batch methods could be used, in particular the classic Bryson-Frazier technique (Bryson and Frazier, 1963). As in this case the cost function is directly minimised, no problem of suboptimality should be encountered, as long as the minimisation is numerically feasible. Computational feasibility may become a serious issue for highly non-linear, multidimensional, noise-affected problems.
3. application of Haupt's two-step estimator. In this case a suitable dynamic model should be the Wiener process, given by (2.144). Some improvement with respect to

the IEKF based on the same assumptions should be possible, as most (if not all) of the information available comes from the measurement equation.

Among the three different choices, the use of an iterative batch technique seems to be the most suitable, even though statements about performance of non-linear estimators are quite difficult and somewhat arbitrary before any simulation is done. In this case, of course, the estimation would be computationally expensive.

In conclusion, it is worth pointing out that the common problem of the "smearing" effect is still present: faults actually represented by deviation of a small number of highly distorted parameters are likely to be identified as deviations of a larger number of performance parameters. This entails that parameters that are actually modified by physical faults are usually underestimated whereas actually undeteriorated components are estimated to be deteriorated as well. The "smearing" effect will be higher if the filter is used to estimate sensor biases as well, since the number of states to be determined is larger.

2.8 Minimum Model Error estimation

As stated in section 2.7, apart from the issue of non-linearity, the coarse modelling of the fault dynamics is likely to reduce the accuracy of ***on-wing diagnostics*** based on Kalman filtering, due to the low robustness of this estimator with respect to unmodelled effects. When the particular problems of optimality of the non-linear KF and the common problems of accuracy of the estimation caused by the "smearing" effect are taken into account as well, other estimation techniques may result more suitable. A novel method showing interesting properties has been introduced and tested recently by Mook (Mook, 1988). The technique is a batch estimator for non-linear, poorly modelled dynamic systems, namely the *Minimum Model Error* (MME) estimator. The method is especially appropriate for post-experiment estimation of dynamic systems whose dynamic modelling equations are suspected or known of containing errors. The approach is different from the KF's one in many respects. Whereas the Kalman filter accounts for uncertainty about the dynamics by means of either a coarse modelling (e.g. constant or Wiener process) or the process noise (assumed to be Gaussian, white and zero mean), the MME estimator quantifies the unavoidable error as a vector that is calculated during the estimation and therefore allows for correction of the model in a deterministic way. The following points can be made about the way the KF deals with modelling errors:

1. The assumption according to which the error is expressed by means of Gaussian, white, zero mean noise is often arbitrary. More often the model error is a smooth function resulting from typical *simplifications (e.g. linearisation, ignoring secondary* effects or higher order terms, or just plain ignorance).
2. The KF can account for model uncertainty by means of parameterisation: the model is defined by a chosen form (e.g. polynomial) and a number of parameters to be determined (the polynomial's coefficients). This can give the KF some capability of estimating the actual dynamics by augmenting the state vector with the unknown parameters (Gelb, 1970). On the one hand prior knowledge about the most likely types of fault's temporal evolution can be exploited, on the other hand if the chosen

model is unsuitable for the actual dynamics the final accuracy will be low. Furthermore, accuracy is doomed to be affected by suboptimality problems, as described in section 2.7 in a greater detail.

The MME estimator relies on the following equations:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t),t) \qquad \text{dynamic model} \qquad (2.145)$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}(t_k),t_k) + \mathbf{v}_k \qquad k = 1,2,...,M \qquad \text{measurement model} \qquad (2.146)$$

The usual assumptions about the measurement noise are accepted: white, Gaussian, uncorrelated, zero-mean noise with covariance matrix R_k.

An optimal estimate is searched for during some specified time interval $[t_0, t_M]$.

The dynamics is known or suspected to be affected by errors and so eq. (2.145) is modified to:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),t) + \mathbf{d}(t) \qquad (2.147)$$

where a to-be-determined unmodelled disturbance vector $\mathbf{d}(t)$ has been added to the known dynamic equation.

The following cost function is then minimised with respect to $\mathbf{d}(t)$:

$$J = \sum_{k=1}^{M} [\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}(t_k),t_k)]^T R_k^{-1} [\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}(t_k),t_k)] + \int_{t_0}^{t_M} \mathbf{d}^T(t) \cdot W \cdot \mathbf{d}(t) dt \qquad (2.148)$$

where the caret "^" denotes the estimated value of the state vector and the weight matrix W is symmetric and has to be determined.

Determination of the weight matrix W is performed according to the ***covariance constraint*** method. The idea is as follows: consistent estimates of the states must match the available measurements with a residual error covariance that is approximately equal to the known measurement covariance matrix. In the common case in which the measurements covariance matrix is constant, the covariance constraint reduces to:

$$\frac{1}{M} \sum_{k=1}^{M} [\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}(t_k),t_k)]^T \cdot [\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}(t_k),t_k)] \approx R \qquad (2.149)$$

The use of an average covariance constraint allows for a reduction of the sensitivity to small sample statistical anomalies.

Minimisation of the first term of (2.148) causes the predicted state estimates to fit the measured data according to our prior knowledge about the different measurement scatter. This is done by weighting the quadratic terms with the known measurement covariance matrix R_k. The second term in (2.148) means that the amount of unmodelled effects, expressed by the vector $\mathbf{d}(t)$, should be kept to a minimum. Actually the

presence of $\mathbf{d}(t)$ in the cost function produces somewhat ambivalent results. If the original dynamic model contains large errors, the addition of a large $\mathbf{d}(t)$ should enable the estimate to better fit the measurements. This causes a reduction of the summation term in the cost function. However, the presence of a large value of $\mathbf{d}(t)$ will produce a large value of the integral. The proper balance between the two kinds of effects depends on the value of W. Since W is determined by the covariance constraint condition, minimisation of the cost function (2.148) leads to the smallest $\mathbf{d}(t)$ which is statistically consistent with the measurements. Basically, if W is too large, the estimate is too far from the measurements, if W is too small the estimate is too close to the measurements. The minimisation of the cost function with respect to $\mathbf{d}(t)$ and subject to the constraint eq. (2.149) can be performed with the Lagrange method. Thus, a new cost function is built:

$$\bar{J} = J + \boldsymbol{\lambda}^T \cdot [\mathbf{f}(\mathbf{x}(t),t) + \mathbf{d}(t) - \dot{\mathbf{x}}(t)] \tag{2.150}$$

where the costate vector $\boldsymbol{\lambda}(t)$ has been introduced. Necessary condition for $\bar{J}$ to be minimum is (Bryson and Ho, 1975):

$$\delta \bar{J} = 0 \tag{2.151}$$

that is the variation of $\bar{J}$ has to be zero for every possible variation of the variables. This leads to the following Two Point Boundary Value Problem (TPBVP):

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),t) + \mathbf{d}(t) \tag{2.152}$$

$$\dot{\boldsymbol{\lambda}}(t) = -\left[\frac{\partial \mathbf{f}(\mathbf{x}(t),t)}{\partial \mathbf{x}}\right]^T \cdot \boldsymbol{\lambda}(t) \tag{2.153}$$

subject to the conditions:

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{2.154}$$

$$\boldsymbol{\lambda}(t_0^-) = 0 \tag{2.155}$$

$$\boldsymbol{\lambda}(t_k^+) = \boldsymbol{\lambda}(t_k^-) + 2H_k^T R_k^{-1} \cdot [\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}(t_k),t_k)] \tag{2.156}$$

$$\boldsymbol{\lambda}(t_M) = 0 \tag{2.157}$$

where:

$$H_k = \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]_{\hat{\mathbf{x}}(t_k),t_k} \tag{2.158}$$

Eq. (2.152)-(2.157) describe a TPBVP for a particular value of the weight matrix W. The correct value of W can be chosen by using a gradient descent technique, for example the classic Gauss-Newton (Mook, 1988). The choice of W can be compared to the tuning of the KF. In the MME estimator, though, the value is obtained when the covariance constraint is satisfied. Even if the TPBVP may be complex to solve, there exist many well-tested numerical techniques able to provide results in reasonable time.

The following advantages have to be highlighted:

- tests have shown that even in case of complete ignorance (i.e. $\dot{\mathbf{x}}(t) = \mathbf{d}(t)$) the algorithm is able to estimate the system dynamics quite accurately
- the estimator is well suitable for use with arbitrary non-linear systems
- the estimator shows robustness with respect to:
  a) low measurement frequency
  b) low measurement accuracy
  c) low total number of measurements
- tuning has a sound physical basis

The main disadvantage is:

- large computational burden.

The list of the estimator's properties given above suggests that the MME estimator should suit the on-wing diagnostic problem better than the linear or even non-linear Kalman filter, provided the computational feasibility is reached for such a complex set of equations. Therefore, an attempt has been made to adapt the MME estimator to on-wing diagnostics.

A straightforward method would be the augmentation of the state vector to include the unknown biases:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ \mathbf{b} \end{bmatrix} \tag{2.159}$$

where

- $\mathbf{x}$ is the performance parameter vector
- $\mathbf{b}$ is the measurement bias vector

The equations necessary for the MME technique are:

$$\dot{\mathbf{X}} = \mathbf{d} \tag{2.160}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{b}_k + \mathbf{v} \tag{2.161}$$

A straightforward application of the method would produce the estimate of both performance parameters and sensor biases.
The drawbacks are:

- the large number of unknowns: if n and m are the number of performance parameters and measurements respectively and three environment and power setting parameters are assumed, the total number of unknown quantities in the consequent TPBVP are $2(n+m+3)$.
- the observability of the system may be low. It obviously depends on the choice of measurements, but problems of distinguishing between performance parameters and sensor bias deviations are likely to occur.

In order to overcome the pitfalls underlined above, a special approach has been developed for on-wing diagnostics. The technique is based on the following ideas:

- a reduction of the number of parameters to be estimated could be achieved if the calculation of the unmodelled disturbance vector $\mathbf{d}(t)$ was somehow exploited to estimate the bias vector
- detection and identification of sensor and engine component faults can be eased if the temporal trends of the various quantities to be determined are properly linked. If a suitable set of measurements is available, i.e. if the measurement set is chosen according to GPA requirements, then the knowledge of the non-linear relations among the trends of the different quantities to be estimated should be exploited in the frame of a non-linear estimation technique such as the MME.

In the light of the points explained above, a two-step MME estimator has been developed.
The first step allows for filtering out of the measurement noise. Various techniques can be employed, but since nothing is known about the temporal evolution of the measurement series, a simple MME estimator seems to be well suited. Thus, the equations to which the common MME estimator is applied are:

$$\mathbf{y}_k = \mathbf{z}_k + \mathbf{v} \tag{2.162}$$

$$\dot{\mathbf{z}} = \mathbf{d} \tag{2.163}$$

The MME algorithm produces an estimate of the measures $\hat{\mathbf{z}}(t)$ and thereby their trends (coincident with the disturbance vector $\mathbf{d}(t)$).
The full non-linear measurement equation containing both performance parameters and sensor biases is:

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{b}_k + \mathbf{v} \tag{2.164}$$

Substituting $\hat{\mathbf{y}}(t)$ for the solution $\hat{\mathbf{z}}(t)$ of the first step for notational convenience, the following equation is obtained:

$$\hat{\mathbf{y}}(t) = \mathbf{h}(\hat{\mathbf{x}}(t)) + \hat{\mathbf{b}}(t) \tag{2.165}$$

The first step of the method provides an estimation of the measurements after filtering out the noise. In this way the desired temporal relationship between sensor biases and performance parameters is obtained. Then eq. (2.165) is differentiated:

$$\dot{\hat{\mathbf{y}}}(t) = \left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]_{\mathbf{x}=\hat{\mathbf{x}}} \cdot \dot{\hat{\mathbf{x}}}(t) + \dot{\hat{\mathbf{b}}}(t) \tag{2.166}$$

and hence:

$$\dot{\hat{\mathbf{x}}}(t) = \left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]^{-1}_{\mathbf{x}=\hat{\mathbf{x}}} \cdot \dot{\hat{\mathbf{y}}}(t) - \left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]^{-1}_{\mathbf{x}=\hat{\mathbf{x}}} \cdot \dot{\hat{\mathbf{b}}}(t) \tag{2.167}$$

The following definitions are given:

$$\mathbf{f}(\hat{\mathbf{x}}(t),t) = \left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]^{-1}_{\mathbf{x}=\hat{\mathbf{x}}} \cdot \dot{\hat{\mathbf{y}}}(t) \tag{2.168}$$

$$N(\hat{\mathbf{x}}(t)) = -\left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]^{-1}_{\mathbf{x}=\hat{\mathbf{x}}} \tag{2.169}$$

It is reminded that $\left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]$ is the Influence Coefficient Matrix used in GPA.

Eq. (2.167) can be re-written as follows:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t),t) + N(\hat{\mathbf{x}}(t)) \cdot \dot{\hat{\mathbf{b}}}(t) \tag{2.170}$$

In the second step it is assumed that (2.170) is the dynamic equation amenable to application of an MME estimator. In this case, though, the dynamics is expressed in terms of a non-normal differential equation. The complete equation set results:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),t) + N(\mathbf{x}(t)) \cdot \dot{\mathbf{b}}(t) \tag{2.171}$$

$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i) + \mathbf{b}_i + \mathbf{v} \tag{2.172}$$

It is worth pointing out two advantages of this approach:

- a reduction of the quantities to be estimated is achieved, since the bias vector is now related to the disturbance vector $\mathbf{d}(t)$ to be determined
- a sensible fault dynamics has been created, based on pre-filtering of the measurement noise and thorough knowledge of the non-linear measurement vector equation.

The cost function to be minimised is in this case:

$$J = \sum_{k=1}^{M} [\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k]^T R^{-1} [\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k] + \int_{t_0}^{t_M} [N(\mathbf{x}) \cdot \dot{\mathbf{b}}]^T \cdot W \cdot [N(\mathbf{x}) \cdot \dot{\mathbf{b}}] dt \tag{2.173}$$

subject to the constraint equation (2.171).
According to Bryson and Ho (1975), the constrained minimisation task can be carried out by adjoining the constraint equation multiplied by the costate vector to the cost function J and minimising the new function so obtained:

$$\bar{J} = J + \boldsymbol{\lambda}^T \cdot [\mathbf{f}(\mathbf{x}(t), t) + N(\mathbf{x}(t)) \cdot \mathbf{b} - \dot{\mathbf{x}}(t)] \tag{2.174}$$

Necessary condition for $\bar{J}$ to be minimum is that the variation be zero:

$$\delta \bar{J} = 0 \tag{2.175}$$

for every possible $\delta \mathbf{x}$.
Once again, the correct value of the weight matrix W is reached when the covariance constraint is satisfied.
Actually the subject of function optimisation covers a very broad area and several different techniques can be used to find an optimal or near-optimal solution. Given the introductory character of this brief foray into on-wing diagnostics, a variational method is sketched that is similar to the standard one used for the common MME. However, further insight should be gained in this area in order to choose the best possible optimisation technique for the problem at hand.
The main difficulty in the minimisation of $\bar{J}$ by means of a variational technique is that in the summation term the bias vector $\mathbf{b}(t)$ whereas in the integral term its derivative $\dot{\mathbf{b}}(t)$ are respectively present. Therefore, a way has to be devised to express the bias vector $\mathbf{b}(t)$ in terms of its derivative $\dot{\mathbf{b}}(t)$ for the straightforward variational technique to apply. This has been done and the detailed treatment is given in appendix G. Minimisation of the constrained cost function is achieved by finding the solution of the following sequence of TPBVPs:

$$\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right] \cdot \dot{\mathbf{x}} = \dot{\mathbf{y}} - \dot{\mathbf{b}} \tag{2.176}$$

$$2\left[\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-1} \cdot \dot{\mathbf{b}}\right]^T \cdot W \cdot \left[\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-2} \left[\frac{\partial^2 \mathbf{h}}{\partial \mathbf{x}^2}\right] \cdot \dot{\mathbf{b}}\right] + \boldsymbol{\lambda}^T \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-2} \left[\frac{\partial^2 \mathbf{h}}{\partial \mathbf{x}^2}\right] (\dot{\mathbf{y}} - \dot{\mathbf{b}}) = \dot{\boldsymbol{\lambda}}^T \tag{2.177}$$

$$2W \cdot \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-1} \cdot \dot{\mathbf{b}} - \boldsymbol{\lambda} = -\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^T \cdot \mathbf{p}_k \tag{2.178}$$

subject to the conditions:

$$\mathbf{x}_0 = 0 \tag{2.179}$$

$$\mathbf{b}_0 = 0 \tag{2.180}$$

$$\boldsymbol{\lambda}^T(t_k^+) = \boldsymbol{\lambda}^T(t_k^-) + 2R^{-1} \cdot \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]_k \cdot [\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k] \tag{2.181}$$

If the weight matrix W is assumed not only symmetric but also diagonal, the algebraic equation (2.178) can be further simplified:

$$\dot{\mathbf{b}} = \frac{1}{2} W^{-1} \cdot \left(\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^T \cdot \boldsymbol{\lambda} - \left(\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^T \right)^2 \cdot \mathbf{p}_k \right) \tag{2.182}$$

The proposed technique should provide the following advantages:

- high diagnostic accuracy: the actual non-linearity of the system is fully retained and both sensor biases and measurement noise are accounted for. An approximate fault dynamics model is created and modified during the estimation process
- the tuning is straightforward: it is achieved by simply satisfying the Covariance Constraint, which has a clear physical meaning
- good robustness with respect to noise, low measurement frequency, low total number of measurements
- dimensionality reduction with respect to the standard MME estimator applied to gas turbine diagnostics
- physical plausibility.

The disadvantages are:

- first and second derivatives of the measurement vector equation have to be calculated or approximated by means of a numerical technique
- the technique is definitely very burdensome from a computational point of view.

Actually the summary about pro's and contra's of the proposed MME technique highlights the great potential for on-wing diagnostics and the numerical problems likely to be encountered.

2.9 Maximum Likelihood estimation

Maximum Likelihood (ML) estimation is widely used for dynamic systems and especially for aircraft dynamics estimation, where the non-linearity can be large and measurements are usually affected by noise and biases. As already shown in section 2.6, dealing with the Bayesian approach, maximisation of the conditional probability density function provides the solution that maximises the probability of occurrence of the state vector, namely the performance parameter vector, given a set of measurements. It can be shown (Jazwinski,

1970) that maximising the probability density function means minimising a cost function which is the common weighted least squares objective function.
The ML estimation, as stated earlier, is an output error technique, able to account for both measurement and model errors. Moreover, it can be used even in non-linear problems. For these reasons, it seems to be suitable for gas turbine diagnostics as well.

The common approach for linear systems is briefly outlined and the necessary modifications for gas turbine diagnostics are suggested.
The following set of equations is given:

$$\dot{\mathbf{x}} = A \cdot \mathbf{x} \qquad \text{dynamic equation} \qquad (2.183)$$

$$\mathbf{y}_i = C \cdot \mathbf{x}_i + \mathbf{v}_i \qquad \text{measurement equation} \qquad (2.184)$$

where:
- $\mathbf{x}$ is the state vector
- $\mathbf{y}$ is the measurement vector
- $\mathbf{v}_i$ is the noise vector.

The usual assumptions about measurement noise are accepted: white, Gaussian, uncorrelated and zero-mean noise.
The elements of the matrix A, describing the dynamics, are assumed to be unknown and have to be estimated.
The following cost function has to be minimised to get the maximum likelihood estimation:

$$J = J(\Theta) = \frac{1}{2}\sum_{i=1}^{N}[\mathbf{y}_i - C \cdot \hat{\mathbf{x}}_i]^T \cdot R_k^{-1} \cdot [\mathbf{y}_i - C \cdot \hat{\mathbf{x}}_i] \qquad (2.185)$$

where:
- Θ is the vector collecting all the elements of the matrix A
- R_k is the measurement noise covariance matrix at time k

Eq. (2.185) highlights that J is a function of the parameters defining the unknown dynamics, i.e. Θ. The minimisation will then be carried out with respect to the elements of the dynamics matrix.
The following definition is given:

$$\mathbf{z}_i = C \cdot \mathbf{x}_i \qquad (2.186)$$

In order to make the minimisation task computationally feasible, the vector $\mathbf{z}_i$ can be expanded in Taylor's series up to the first order:

$$\mathbf{z}_i(\Theta) \cong \mathbf{z}_i(\Theta_0) + \left.\frac{\partial \mathbf{z}_i}{\partial \Theta}\right|_{\Theta_0} \cdot \Delta\Theta \qquad (2.187)$$

The quantities

$$\left.\frac{\partial z_i}{\partial \theta_j}\right|_{\Theta_0} \tag{2.188}$$

are called sensitivities and are usually calculated in either of the two following ways:

- by integrating the sensitivity equations
- with a numerical method such as finite differences.

In both methods most of the computing time is needed just to evaluate the sensitivities. Therefore, techniques have been devised, which are more efficient computationally, in order to ease and speed up the estimation procedure. In particular the Modified Newton Raphson with Estimated Sensitivities (MNRES) method has shown good performance (Murphy, 1984). In this method the sensitivities are approximated as slopes of a surface which is fitted to a set of points. Depending on the degree of accuracy and the computing time required, the kind of surface and fitting method are chosen accordingly. The final estimation accuracy is the same as that for the common MNR technique and the computing requirements (memory and time) are definitely lower.

It is worth noting that the approximation expressed by eq. (2.187) makes the cost function (2.185) quadratic and this remarkably simplifies the minimisation task. The actual effectiveness of the approximation for highly non-linear problems has to be tested, even though there exist many examples in literature where this choice proved to be successful.

For J to be minimum, the following condition is necessary:

$$\frac{\partial J}{\partial \Theta} = 0 \tag{2.189}$$

It should be noted that an unconstrained minimisation task has to be performed.

A Modified Newton Raphson (MNR) technique can be used to find the solution of the above equation.

The following definition is given:

$$\mathbf{v}_i = \mathbf{y}_i - \hat{\mathbf{z}}_i(\Theta_0) \tag{2.190}$$

Manipulation of the eq. (2.189) provides:

$$\Delta\Theta = -M^{-1} \cdot \left.\frac{\partial J}{\partial \Theta}\right|_{\Theta_0} \tag{2.191}$$

where:

$$M = \sum_{i=1}^{N} G_i^T \cdot R^{-1} \cdot G_i \tag{2.192}$$

is the so-called Fisher matrix and

$$\left.\frac{\partial J}{\partial \Theta}\right|_{\Theta_0} = -\sum_{i=1}^{N} G_i^T \cdot R^{-1} \cdot \mathbf{v}_i \tag{2.193}$$

The iterative process is given by:

$$\hat{\Theta}_{k+1} = \hat{\Theta}_k + \Delta\hat{\Theta}_{k+1} \tag{2.194}$$

The new point $\hat{\Theta}_{k+1}$ is substituted for the point giving the maximum $J(\Theta)$. Convergence is reached when $\Delta J / J$ and $\Delta\Theta / |\Theta|$ are small enough.

When convergence is achieved, eq. (2.183) can be integrated and $\mathbf{x}$ calculated. It is worth noting that even though the estimation technique may seem too easy for complex real world problems, it is currently being successfully applied to many real systems characterised by significant non-linearities.

The above treatment can be adapted to account for both system's non-linearity and biased measurements. Two different approaches are proposed here.

An augmented state vector can be introduced which contains both the performance parameters to be estimated and the unknown measurement biases:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ \\ \mathbf{b} \end{bmatrix} \tag{2.195}$$

Either a simply linear or a more complex dynamic equation can be chosen. In the sequel the easiest approach is shown. The measurement equation becomes:

$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i) + \mathbf{b}_i + \mathbf{v} \tag{2.196}$$

and can be re-written as follows:

$$\mathbf{y}_i = \mathbf{f}(\mathbf{X}_i) + \mathbf{v} \tag{2.197}$$

whereas the dynamics is simply represented by:

$$\dot{\mathbf{X}} = C \cdot \mathbf{X} \tag{2.198}$$

The common ML method, where the sensitivities are calculated according to the MNRES technique, can be applied.

The main drawbacks of the technique outlined above are:

- the large number of parameters to be estimated: if the performance parameters and the sensor biases are for example 20, the matrix C is made of 400 elements, all of which have to be estimated

- the observability of the system, once the measurement set is fixed, is likely to be low.

In the light of the above-mentioned pitfalls, the ML estimation has been further developed for use with gas turbine diagnostics.
First the measurement noise is filtered out by means of a non-linear estimation technique applied to the following measurement equation:

$$\mathbf{y}_i = \mathbf{w}_i + \mathbf{v} \tag{2.199}$$

Even though ML could be used, MME seems to be more suitable due to its capability of dealing with non-linearity. As already done with the MME developed for diagnostics, knowledge of the estimated trends of the measurements is useful to study the relationship between the performance parameters and the measurement biases. The same passages as for the MME are done:

$$\hat{\mathbf{y}}(t) = \mathbf{h}(\hat{\mathbf{x}}(t)) + \hat{\mathbf{b}}(t) \tag{2.200}$$

$$\dot{\hat{\mathbf{y}}}(t) = \left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]_{\mathbf{x}=\hat{\mathbf{x}}} \cdot \dot{\hat{\mathbf{x}}}(t) + \dot{\hat{\mathbf{b}}}(t) \tag{2.201}$$

$$\dot{\hat{\mathbf{x}}}(t) = \left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]^{-1}_{\mathbf{x}=\hat{\mathbf{x}}} \cdot \dot{\hat{\mathbf{y}}}(t) - \left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]^{-1}_{\mathbf{x}=\hat{\mathbf{x}}} \cdot \dot{\hat{\mathbf{b}}}(t) \tag{2.202}$$

The following definitions are given:

$$\mathbf{f}(\hat{\mathbf{x}}(t),t) = \left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]^{-1}_{\mathbf{x}=\hat{\mathbf{x}}} \cdot \dot{\hat{\mathbf{y}}}(t) \tag{2.203}$$

$$N(\hat{\mathbf{x}}(t)) = -\left[\frac{\partial \mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}\right]^{-1}_{\mathbf{x}=\hat{\mathbf{x}}} \tag{2.204}$$

Eq. (2.202) can be written as follows:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t),t) + N(\hat{\mathbf{x}}(t)) \cdot \dot{\hat{\mathbf{b}}}(t) \tag{2.205}$$

The ML estimator is then applied to the following set of equations:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),t) + N(\mathbf{x}(t)) \cdot \dot{\mathbf{b}}(t) \tag{2.206}$$

$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i) + \mathbf{b}_i + \mathbf{v} \tag{2.207}$$

where the biases are parameterised to be able to track any likely sensor bias temporal evolution (slow drift, sudden step, parabolic increase, etc.). If M is the number of

measurements and P parameters are used for each measurement, the cost function is minimised with respect to the $(M \cdot P,1)$ vector:

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ . \\ . \\ . \\ \mathbf{a}_m \end{bmatrix} \tag{2.208}$$

The cost function becomes:

$$J = J(\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{N}[\mathbf{y}_i - (\mathbf{h}(\mathbf{x}_i) + \mathbf{b}_i)]^T \cdot R^{-1} \cdot [\mathbf{y}_i - (\mathbf{h}(\mathbf{x}_i) + \mathbf{b}_i)] \tag{2.209}$$

As far as the parameterisation of the bias evolution with time is concerned, statistical knowledge of the more likely kinds of fault could be used. Simple polynomial or spline functions should be sufficiently accurate. It is worth noting that usual applications of the ML especially for aircraft dynamics studies require the calculation of several quantities. This means that a large number of parameters (even 10) could be used for each measurement in order to approximate the bias functions accurately. Once the biases' parameters have been estimated, the performance parameters are calculated by integration of the dynamic equation (2.206). The integration should represent the most burdensome phase in the method, especially because of the non-linearity involved.

In conclusion, even though non-linearity and sensor biases make the estimation of gas turbine performance difficult, ML estimation is a well-defined and suitable technique, which should be tested in order to ascertain its actual capabilities. The issue about computational feasibility should be analysed as well. Nonetheless, this estimation technique should be better suited than the EKF or IEKF for gas turbine performance estimation.

CHAPTER 3

NEURAL NETWORKS AND GAS TURBINE DIAGNOSTICS

3.1 Introduction

As pointed out in the previous chapter, many sources of uncertainty affect the accuracy of gas turbine diagnostics. The complexity of the problem at hand has led to the application of a wide range of techniques trying to cope with the various real-world effects and then make GPA effective.
Recently a great deal of research has been done to apply Artificial Intelligence (AI) methods to diagnostics. In particular Computational Intelligence (CI) techniques, called Neural Networks (NNs), show interesting properties and are currently used for diagnostic tasks.
The amount of uncertainty affecting the gas turbine diagnostic process and the large number of available NN-based methods make the application of NNs to gas turbine diagnostics rather complex.

In the sequel a brief introduction to NNs is given, together with a short literature review about NN-based engine diagnostics. Then attention is paid to the measurement uncertainty issues and how NNs can tackle them. Two works done by the author in the area are briefly described and eventually an evaluation of the suitability of NNs for gas turbine fault diagnosis is given.

3.2 An introduction to Neural Networks

NNs (or Artificial Neural Networks, ANNs) differ from conventional estimation techniques in many respects. However, the main difference is that the latters rely on a mathematical model of the process to be analysed (hence they are called model-based), while the formers learn from examples. NNs can actually be used when there exists no model to describe the physical phenomenon under analysis or the model itself is either too poor or too complex to be utilised. Obviously, data are necessary to enable the nets to glean useful information. Data can be experimental or simulated. As a matter of fact, even when an accurate model exists usage of NNs can help tackle a wide range of problems due to inherent capabilities of these CI techniques.

NNs can be defined as parallel distributed processors able to store knowledge as experience and make it available for use.

The following definitions apply to the most common kind of nets, the so-called feedforward backpropagation neural nets. However, most of them apply to other widespread neural techniques as well.

- The network is made of units called *neurons*, each performing a weighted sum of its own inputs. The sum is then passed through a function, the so-called *activation function*, usually non-linear. The output of the j-th neuron is:

$$y_j = \varphi_j(\sum_{i=0}^{N} w_{ji} \cdot x_i) \tag{3.1}$$

  where φ_j is the activation function applied to the weighted sum of the inputs (x_i).
- Interneuron connections called *weights* (w_{ji} as in (3.1)) are used to store the knowledge.
- Knowledge is acquired by the network through a learning process (*learning* or *training phase*), during which proper algorithms change the values of the weights.

When training is over, the weights are fixed and the net can be used in the so-called recall mode.
Neurons are usually grouped by layers depending on the kind of architecture and learning algorithm.

Some NN properties are the following:

- they can extract useful information from examples used during training: they are particularly suitable to tackle problems for which an exact algorithmic solution does not exist, but a large number of examples are available
- they generalise from examples: they can produce an acceptable answer even to previously unseen data (*generalisation property*)
- they are able to extract basic information even from noisy data
- they are self-learning methods, since correct answers can be achieved even by using no prior knowledge about the physical process to approximate
- they can adapt themselves to changing operating conditions (learning on the job, or in situ)
- whereas the learning phase is often long and computationally expensive, the speed of response of the network used in recall mode is very high
- they are inherently non-linear
- they can perform data fusion: different kinds of data (vibrational, thermodynamic, electrostatic data for a gas turbine) could be used altogether to produce an answer, even though no such comprehensive theoretical model exists.

Data can be obtained experimentally or by a numerical simulation and the consequent development of diagnostics using NNs is usually different, because of the different quantity and quality of the data.

3.2.1 Backpropagation

Among the various neural algorithms, backpropagation is one of the most common, simple and effective ones to train feedforward NNs (Haykin, 1994; Hassoun, 1995). Due to its wide use, it is worthwhile to briefly describe it. Moreover, understanding the way this simple net works can help introduce more complex neural structures later on.

The typical feedforward net, also named Multi-Layer Perceptron (MLP), is made of three layers: input, hidden and output layer. Each layer is made of a set of neurons. For hidden and output neurons the output or activation value is computed according to (3.1), where,

- y_j is the output of neuron j in the current layer
- w_{ji} is the weight going from neuron i in the preceding layer to neuron j in the current layer
- N is the number of neurons in the preceding layer
- x_i is the output of neuron i in the preceding layer
- a bias contributes to the so-called *net value* (i.e. the weighted summation of inputs used as argument of the activation function): x_0 is usually assumed to be equal to -1.

For input neurons, usually the net value coincides with the corresponding input, the activation function is the identity and no bias is used. In this case input neurons are not processing nodes but simply input nodes. Sometimes, though, the activation function is non-linear and then the input and output values of input neurons differ.
In the typical architecture every neuron is connected through weights to all neurons of the succeeding layer and has no connection with other neurons in the same layer (intralayer weights). Direct connections between non-adjacent layers are allowed but seldom used.
For a certain value of the set of weights, when the net is presented with an input pattern, the neurons' output values are computed layer by layer, from the input to the output one and in this way an output pattern is produced. The name "feedforward" stems from the absence of any feedback when the net is used in recall mode. Fig. 3.1 shows a single hidden layer feedforward net with 8 input, 6 hidden and 4 output neurons. The arrows show the direction of the information flow. Biases are not represented.

The simplest way to train a feedforward net like the one displayed in fig. 3.1 is through a learning algorithm named ***backpropagation***. Data used for training are pairs of input-output vectors. Output vectors are also called target as objective of the training is to modify weights to force the net to reproduce them whenever presented with the corresponding input patterns. Main utilisation of a net of this kind is to carry out pattern recognition or, in mathematical terms, a multi-dimensional approximation. *Training set* vectors are used by the algorithm to modify the weights, whereas *test set* vectors are not involved in this process but are used to test the net generalisation properties.

The training is made of 2 steps:
1) *feedforward phase*: given the chosen activation function, for the input vector the weighted sums of the neurons are calculated according to (3.1), layer by layer, from the

input to the output layer. The output values are then compared with the target values and errors are calculated

2) *backpropagation phase*: errors are used by the learning algorithm to change the values of the weights layer by layer, from the output to the input layer.

These two-step calculations are repeated for every example of the training set.

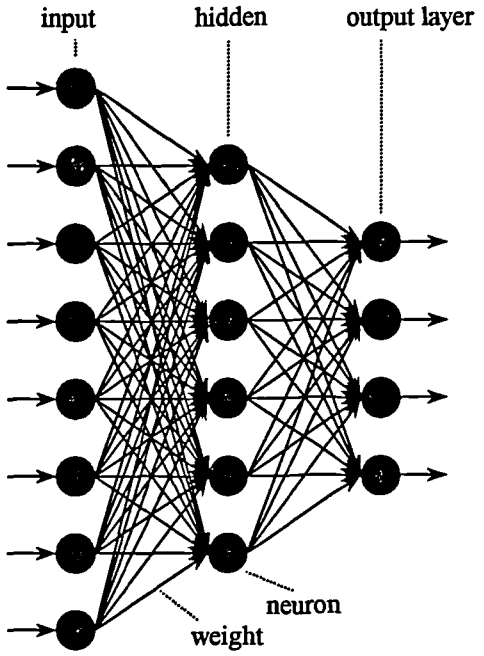


Fig. 3.1: a feedforward neural net

If *pattern-by-pattern training* is performed, the sequence of pattern presentation is random and the two-step computations described above are done after presentation of each pattern. Application of the two step procedure to the whole training set makes up a so-called *epoch.*

If *batch training* is performed, the whole training set is used to calculate an average error and then the backpropagation step is carried out. Usually the pattern-by-pattern training is preferred due to the better generalisation performance achievable.

A rather detailed description of backpropagation is given here, because the algorithm has been modified to better suit the sensor validation problem, as will be shown in section 3.7.

For the output neuron j at presentation of the n-th vector (the so-called *iteration*), the error is:

$$e_j(n) = d_j(n) - y_j(n) \tag{3.2}$$

where:

- $d_j(n)$ is the j-th target output

- $y_j(n)$ is the j-th calculated output.

If the network has p output neurons, the *instantaneous sum of squared errors* at the n-th iteration is defined as follows:

$$E(n) = \frac{1}{2}\sum_{j=1}^{p} e_j^2(n) \tag{3.3}$$

where $E(n)$ is a function of the weights of the network.
A gradient-descent technique is used to change the weights:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{3.4}$$

where the following definition applies:

$$\Delta w_{ji}(n) = w_{ji}(n) - w_{ji}(n-1) \tag{3.5}$$

η is a constant called *learning rate parameter* and is set at the beginning of training.
The derivative of the error present in eq. (3.4) can be properly developed for implementation and its expression is different for output and hidden neurons.

a) output neurons

According to the chain rule of differentiation:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{3.6}$$

where $v_j(n)$ is the net value of neuron j:

$$v_j(n) = \sum_{i=0}^{M} w_{ji}(n) \cdot x_i(n) \tag{3.7}$$

with M equal to the number of neurons of the preceding layer.
Every factor in the R.H.S. of (3.6) can be properly expressed:

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \tag{3.8}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{3.9}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \tag{3.10}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{3.11}$$

Eq. (3.4) becomes:

$$\Delta w_{ji}(n) = \eta \cdot e_j(n) \cdot \varphi_j'(v_j(n)) \cdot y_i(n) \tag{3.12}$$

The *local gradient* is defined for neuron j as follows:

$$\delta_j(n) = e_j(n) \cdot \varphi_j'(v_j(n)) \tag{3.13}$$

Accordingly, (3.12) can be written:

$$\Delta w_{ji}(n) = \eta \cdot \delta_j(n) \cdot y_i(n) \tag{3.14}$$

b) hidden neurons

Eq. (3.6) can also be written as follows:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{3.15}$$

In this case, $\frac{\partial E(n)}{\partial y_j(n)}$ has to be adequately developed:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_{k=0}^{p} e_k(n) \cdot \frac{\partial e_k(n)}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)} \tag{3.16}$$

where p is the number of neurons in the succeeding layer (e.g. the output layer).
The last two factors can be calculated as follows:

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \tag{3.18}$$

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{3.19}$$

According to the definition of local gradient given by (3.13):

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_{k=0}^{p} \delta_k(n) \cdot w_{kj}(n) \tag{3.20}$$

For hidden neurons the following definition of local gradient is given:

$$\delta_j(n) = \varphi_j'(n) \cdot \sum_{k=0}^{p} \delta_k(n) \cdot w_{kj}(n) \tag{3.21}$$

Thus, (3.4) again reduces to (3.12).
The learning rate parameter η is fixed. A low η produces a smoother trajectory but a slower weight adjustment; a large η may introduce instability in the learning curve (average error vs. number of epochs).
A large number of modifications have been applied to the bare algorithm just described (and called standard backpropagation). Among them, one of the most straightforward is the addition of a momentum term, introduced to increase stability and speed of learning:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} + \alpha \Delta w_{ji}(n-1) \tag{3.22}$$

where α is a constant chosen at the onset of training.
Necessary condition for the algorithm to apply is that the activation function φ_j's derivative exists.
Typically, sigmoidal functions are used. A classic choice is the logistic function (fig. 3.2):

$$\varphi_j(v_j(n)) = \frac{1}{1+e^{-v_j(n)}} \tag{3.23}$$

or the hyperbolic tangent (fig. 3.3):

$$\varphi_j(v_j(n)) = \frac{1-e^{-v_j(n)}}{1+e^{-v_j(n)}} \tag{3.24}$$

If the average error is introduced:

$$\overline{E} = \frac{1}{N} \sum_{n=1}^{N} E(n) \tag{3.25}$$

where N is the number of training patterns, the backpropagation algorithm can be regarded as a first order approximation of the maximum descent trajectory in the weight space, which would be obtained by minimising $\overline{E}$ instead of E. Indeed, backpropagation learning is an application of a statistical method known as *stochastic approximation* (Gelb, 1974; Haykin, 1994).

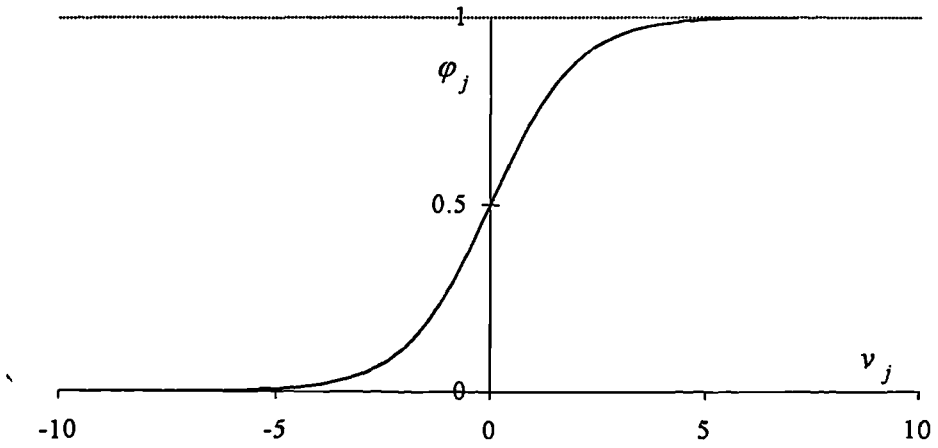


Fig. 3.2: logistic activation function

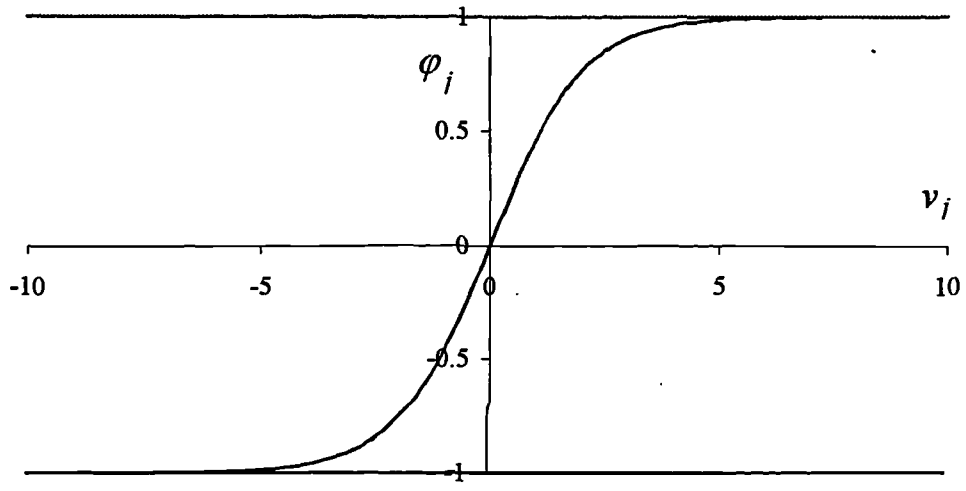


Fig. 3.3: hyperbolic tangent activation function

A number of points have to be made to better understand backpropagation feedforward NNs.

Generalisation. The net's capability to provide accurate answers even when presented with patterns not used for training can be related to the following factors:
a) *Size of the net*: while the nature of the problem dictates the number of input and output neurons, the number of hidden layers and corresponding neurons are design parameters. Usually an increase in the number of hidden neurons decreases the training set error, but a very complex net can "learn too well" the relationship between input and output training set patterns, to such an extent that even noise is recorded (*overfitting*). Such a detailed learning of the training set reduces the generalisation performance. Therefore, a suitable number of hidden neurons is the minimum number according to the

complexity of the process to approximate. There is no defined rule as to the choice of the number of hidden neurons. A trial and error process has to be performed by checking both training and test set errors.
b) *Level of training*: a very large number of epochs may lead to a very low training set error and a large and ever increasing test set error (*overtraining*). Training should be stopped when the test set error is steady or starts increasing.
c) *Quality of the data*: due to the data-driven characteristic of NNs, data must be of good quality in that they must be sufficiently representative of the process to approximate. On the one hand, if data are noisy a larger number of patterns should be used to allow the net to extract the information necessary to approximate the mapping. On the other hand, training the net with noise-free data will definitely worsen the generalisation performance when real (and hence noisy) data will be used.
d) *Quantity of the data*: even though there exist some attempts to establish criteria to set the number of training patterns (e.g. the Widrow rule, see Haykin, 1994), no defined rule is universally accepted. However the quantity of patterns is somewhat related to the complexity of the input-output relationship and the quality of data.

Initialisation. Before training is started, the weights must be given a value. Two different approaches can be pursued:
a) Supervised initialisation: in certain cases hints can be obtained by statistical analysis of the data. This usually results in setting adequate ranges for random generation of weights. For particular problems, the net's structure itself can be chosen with supervision. The corresponding net will not usually be fully connected but is likely to show some kind of modularity. In that case, the weights can be assigned precise values to ease the following training.
b) Random initialisation: weights are given random values uniformly distributed inside a range. The distribution is random in order to reduce the probability of the net getting trapped in a local minimum. This problem seriously affects the performance of MLPs and the usual means to try to overcome it are the randomness of the initialisation and, whenever possible, use of prior knowledge of data and process to approximate in order to tailor the net's design.
Choice of a tight range can reduce the probability of occurrence of a phenomenon called *premature convergence*. This results in saddle points or even large plateaus in the learning curve. Because of premature convergence, training can be substantially lengthened. Premature convergence occurs when the target and calculated values for a significant number of output neurons are very far from one another and the calculated one is close to one of the activation function's asymptotes. If so, the corresponding weight adjustments should be large as well. However, as shown by (3.13), the changes applied to the weights are relatively small and then the adjustment process is slow. The problem can be overcome by choosing a small range for initialisation and a small number of hidden neurons.

Universal approximation theorem. Backpropagation-based NNs are widely used because of the simplicity and performance they can provide. From a theoretical point of view, the success of this kind of nets can be explained through an existence theorem, which is stated below.

Let $\varphi(.)$ be a non-constant, bounded and monotone-increasing continuous function. Let I_p denote the p-dimensional unit hypercube $[0,1]^p$. The space of continuous functions on I_p is denoted by $C(I_p)$. Then, given a function $f \in C(I_p)$ and $\varepsilon > 0$, there exists an integer M and sets of real constants α_i, θ_i and w_{ij}, where $i = 1,\ldots,M$ and $j = 1,\ldots,p$ such that we may define:

$$F(x_1,\ldots,x_p) = \sum_{i=1}^{M} \alpha_i \cdot \varphi\left(\sum_{j=1}^{p} w_{ij} \cdot x_j - \vartheta_i\right) \tag{3.26}$$

as an approximate realisation of the function $f(.)$; that is:

$$\left|F(x_1,\ldots,x_p) - f(x_1,\ldots,x_p)\right| < \varepsilon \tag{3.27}$$

for all $(x_1,\ldots,x_p) \in I_p$.

As the theorem's hypotheses on the function $\varphi(.)$ and the form of the approximation (3.26) are satisfied by the MLP, for every function $f(.)$ it is possible to find a single hidden layer feedforward net able to approximate it uniformly. Although the theorem provides a theoretical confirmation of the effectiveness shown by this kind of net in practical applications, it is of no use in the design of the net itself (i.e. number of hidden neurons). Furthermore, given a certain $f(.)$, the number M of hidden neurons necessary to get the required accuracy ε can be too large for the training to be feasible.

The backpropagation is the simplest and most used learning algorithm. It suffers, though, from many drawbacks and therefore several modifications of the original algorithm have been devised and tested.

The main pitfalls are summarised below.

Local minima: as stated earlier, pattern-by-pattern backpropagation training realises an approximation of the maximum descent trajectory on the average error function $\overline{E}$ surface in the weight space. Occurrence of local minima can lead to suboptimal solutions and then inaccurate prediction.

Slow rate of convergence: it can be regarded as the effect of two distinct causes:

a) Let the error surface be fairly flat along a weight dimension. Thus the corresponding derivative of the error surface is small in magnitude and then the adjustment applied to the weight is small as well. Consequently, many iterations of the network may be required to produce a significant reduction in the error performance of the network.

Let now the error surface be highly steep along another weight dimension. The corresponding derivative of the error surface is large in magnitude. The weight adjustment may be so large that the algorithm overshoots the minimum of the error surface.

b) The direction of the negative gradient vector may point away from the minimum of the error surface. Hence the adjustments applied to the weights may induce the algorithm to move in the wrong direction.

A rather straightforward method to increase the rate of convergence of backpropagation is to apply learning rate adaptation according to some heuristics. These heuristics can be summarised as follows:

1. Every adjustable network parameter of the cost function should have its own individual learning rate.
2. Every learning rate parameter should be allowed to vary from one iteration to the next.
3. When the derivative of the cost function with respect to a weight has the same algebraic sign for several consecutive iterations, the learning rate parameter for that particular weight should be increased.
4. When the algebraic sign of the derivative of the cost function with respect to a particular weight alternates for several consecutive iterations, the learning rate parameter for that weight should be decreased.

An algorithm realising these heuristics is the ***delta-bar-delta*** learning rule, which is detailed in appendix H. From a practical point of view, this training algorithm should allow a higher rate of convergence than the standard backpropagation. As a matter of fact, utilisation of this algorithm has substantially fostered the achievement of the training goals in two different works, which are described in sections 3.5 and 3.6.

3.3 NNs applied to engine diagnostics

The features of NNs, briefly outlined in section 3.2, make them amenable to application to diagnostic tasks. In particular:

- the large level of noise affecting measurements in gas turbines could be coped with by NNs
- even though some parameters have to be chosen at the design phase and at the beginning of training, there is no such critical choice as the one required in the KF-based techniques to set the standard deviations of the performance parameters
- NNs could be trained on-line to monitor the engine's health in real time
- NNs should be able to handle the relatively large non-linearity characterising the relationship between measurements and performance parameters
- NNs could be used to perform data fusion for gas turbine diagnostics: vibrational, aero-thermodynamic, gas path debris data could represent a comprehensive input to a NN-based system.

In the past few years, NNs have been used to monitor and diagnose faults of a wide range of processes: gas turbines, rocket engines, industrial plants, etc. In the sequel an outline of the most interesting applications is given, with particular attention to the possible development of detailed, real world diagnostics of gas turbines.

3.3.1 NASA's approach

Since their introduction an extensive use of NNs has been made by NASA for assessing the health of the Space Shuttle Main Engine (SSME). Even though the SSME is very

different from common gas turbine engines, it is interesting to summarise features and achievements of a neural diagnostic technique developed by NASA (Whitehead et al., 1990a, 1990b). The engine is analysed in its steady state condition. The occurrence of a fault produces variations in the measured quantities and each fault is characterised by a particular temporal pattern of the measurement variations. The diagnostics is carried out by means of a modular neural technique made of three phases:

1. data compression (see also Carr and Cowley, 1995). A backpropagation net is used to carry out Auto-Association.
2. hypersurfaces are created in the compressed data space, each one representing a particular fault. Each hypersurface is given a thickness in order to take the measurement noise into account. The probability of occurrence of a particular fault is provided by the distance of the operating point considered from the hypersurface
3. a backpropagation net performs the final classification task. The net is built to carry out a differential diagnosis: the possible hypotheses of fault are paired off and the net is able to recognise even unknown malfunctions. This is obtained by designing a backpropagation net that is not fully connected: the weights are chosen in order to use the prior knowledge about the nature of the data.

The diagnostic results are excellent: the system is able to promptly detect known faults and classify unknown faults as such.

The following remarks about the use of NNs for diagnostics have to be done in the light of NASA's technique:

- NNs are very well suited to classification tasks. As even GPA's diagnostic answers are reverted to practical recommendations about which component is responsible for the loss of performance, the qualitative classification properties of NNs seem to be very attractive
- even though NNs can generalise, an inherent drawback of these techniques is that they produce good diagnostic answers only when provided with faults not very different from those used during training. If the fault is actually very different, the net will produce a wrong answer, usually as a combination of the faults used during training. Basically, since they are self-learning techniques, they can learn only what they are taught. This lack of flexibility is definitely a disadvantage with respect to model-based techniques. This problem can be overcome by using a large database containing a large number of faults. However, two issues have to be considered carefully:

  1. the presence of a faulty sensor has to be taken into account in some way, because it is not unlikely and can completely impair the diagnostic accuracy

  2. the size of the necessary nets can become excessive (too many neurons) and thereby make the problem difficult to handle given the current computing resources.

  However, NASA's approach allows accounting for unknown faults. This is due to the careful, supervised design of the net and the qualitative approach pursued.
- the use of prior knowledge in the design phase of the net can be very effective. Nonetheless, due to the way the nets work (especially the hidden neurons, when present), exploiting the knowledge of the process to approximate is very difficult.

3.3.2 Eustace's approach

A first application of NNs to gas turbine diagnostics has been provided by Eustace (1993). A diagnostic system has been devised to detect six types of faults (apart from the normal condition of operation) on a GE F404 turbofan engine. Five measurements are used and the task is classification. Input measurements are subject to filtering to cancel out noise and outliers. A simple small backpropagation net is used with 5 input, 6 hidden and 6 output neurons.
The occurrence of a fault should make a single output neuron fire. The database is made of only 70 patterns because data are experimental. The following remarks can be made about Eustace's technique:

- the good diagnostic performance confirms the suitability of NNs for classification
- a small number of faults are detectable, because the database used for training is small. The reason is that data come from a real engine and not from a simulation program
- NNs perform well with real world data
- NNs seem to be more suitable than Expert Systems (especially Knowledge Based Systems, see Vivian and Singh, 1995) for classification
- the problem of rejecting unknown data is particularly serious if real data are used, as they are necessarily few
- if the pattern to be classified is on the border between two different fault conditions, then the difficulty of classification will result in a value of the fired output neuron close to 0.5 as 1 and 0 are respectively the fault and no fault conditions. Similar values will be produced by the other output neurons. This provides a sort of level of confidence of the classification and in case suggests the need for further investigation, possibly by means of other diagnostics techniques.

A similar technique has been used for fault diagnosis of a fleet of engines (Eustace and Merrington, 1995). In this case the problem is even more difficult because of the unavoidable differences among engines of the same kind (engine-to-engine scatter). Baselines from 130 engines have been used and a linear relationship has been detected between the measurements used for diagnosis and three measurements (fuel flow, compressor inlet total temperature, and compressor outlet static pressure). The faults are assumed to be simply given by superimposition on the engine baseline. The input quantities are then the distances from the engine baseline (5 measurements) and the output neurons are 6, representing the no-fault and 5 fault conditions. A Probabilistic Neural Network (PNN) is used which is made of the input layer, a normalisation layer, a hidden layer and an output layer. The net is basically the neural implementation of the Bayesian theory of conditional probability. The hidden layer is made of 60 time 6 neurons, as 60 engine where used for training the net to detect 6 faults. Data from 70 engines have been used to test the net. Remarks about the net are the following:

- the classification performance is good (87,1%), especially when compared with the usual threshold test that provided only 71,2% of correct classification
- the diagnostic task is difficult, as fault areas are actually partially overlapping
- PNNs are not suitable to deal with large scale problems due to the increase in the number of hidden neurons

- the engine-to-engine variations can be taken into account.

Basically the simple approach chosen by Eustace shows the suitability of NNs for real world diagnostics through classification.

3.3.3 The Italian approach

Various neural techniques have been used by Torella and Lombardo (1995) to carry out gas turbine diagnostics. Adaptive Resonance Theory 1 (ART1), Counterpropagation Nets (CP) and simple backpropagation have been used. While ART1 and CN have shown their inherent limitations, the backpropagation resulted well suitable for diagnostics. Classification of 8 different kinds of faults, simply represented by variation of only one performance parameter, has been achieved with a backpropagation net with 17 input measurements. A wide range of numbers of hidden neurons have been used (4 through 51) to test the variation of the net's behaviour. A larger number of hidden neurons provide better generalisation. The following points have to be made regarding this application:

- the classification performance achieved was good
- the net shows robustness with respect to measurement errors: even if a sensor bias was present a correct diagnostic answer has been obtained
- the large number of measurements used as inputs simplifies the classification task. This betters the robustness as well.

Even though the diagnostics developed by Torella is somewhat simplistic, it again shows the good performance of classification achievable by means of NNs.

3.3.4 The German approach

As already seen in section 2.4, an interesting piece of work has been done by Lunderstaedt's team (Junk and Lunderstaedt, 1995). The German research team used NNs for distinguishing between sensor and engine component faults. The diagnostic system performed Sensor Failure Detection, Identification and Accommodation through a modular neural structure. Details about this technique can be found in section 2.4. The following points can be made:

- neural networks and conventional model-based estimation techniques (optimisation algorithms) are used altogether to take advantage of the different characteristics of the methods
- NNs are used to produce the values of the measurement deltas given the performance parameter deltas. In this case NNs are used for providing a quantitative answer
- the following uncertainties are taken into account:
  a) measurement noise
  b) sensor bias
  c) model uncertainties due to the approximate knowledge of the component characteristics
- the second neural module is able to distinguish between sensor bias and engine module faults

- backpropagation is used
- the system's detection and isolation capability are good, whereas accommodation results more difficult. While sensor biases, i.e. systematic sensor errors, are accommodated, slow drifts of the measurements are accommodated only unprecisely: an offset remains

3.3.5 The Greek approach

Kanelopoulos et al. (1997) have done a work which is interesting as it tries to address the issue of the best possible application of NNs to gas turbine diagnostics and modelling as well.

NN-based modelling of gas turbine performance is attempted for two main reasons:

- using a net or even a set of nets in recall mode (when training and testing are completed) requires much less computational effort than using a full aero-thermodynamic engine model. In the latter case a set of highly coupled non-linear equations has to be solved iteratively, whereas the computations required to use the nets in recall mode are very straightforward
- availability of real engine data allows to easily adjust the NN to simulate the performance of a particular engine.

The nets used for modelling a single shaft industrial gas turbine are 4-7-7 feedforward architectures trained by backpropagation. The logistic activation function is used. Input quantities are ambient pressure and temperature, spool speed and power. Output quantities are compressor delivery pressure and temperature, turbine exit temperature, mass flow, fuel flow, compressor and turbine isentropic efficiencies. Data used for training and testing are obtained by simulation. Measurement noise is also simulated and superimposed. Two different cases are considered:

- modelling at fixed nominal rotational speed. The actual spool speed differs from the nominal one just by measurement noise. Several spool speed and inlet temperature-defined operating points are considered. The prediction error is usually below 0.5%. The largest errors are found at the boundary of the area defined by the training patterns.
- modelling along a working line. The same observations as for modelling at a fixed nominal rotational speed can be made.

An important outcome of the study on NN-based modelling is that a set of nets is necessary to cover the entire operational envelope. The performance achievable by using a single net for the whole map is definitely poor. A modular approach allows reaching much better accuracy.
However, NNs are very unlikely to replace aero-thermodynamic models because of accuracy and reliability issues. Furthermore, availability of powerful yet cheap computational platforms and refinement of the iterative methods of solution of the aero-thermodynamic equations strongly attenuate the NN advantage based on the shorter computational time required.

The application of NNs for proper diagnostics of gas turbines confirms the need for a modular approach to tackle even relatively simple problems.

3 feedforward backpropagation NNs are used in series to carry out diagnostics, whose aim is to distinguish between single engine component and single sensor faults for a single shaft gas turbine and provide a number quantifying the severity of the fault occurred.
The features of the proposed method are the following:

- measurements coming from the engine are fed in an aero-thermodynamic model, which calculates the values of the performance parameters (efficiency and flow function of the main engine components). A fault in an engine component will affect just the corresponding performance parameters, while a sensor fault will produce a typical fingerprint in the set of performance parameters. The percentage variations of the performance parameters relative to their values in a nominal engine are used as input to the NN modular system.
- A first net classifies the input performance parameter deltas to find out whether there is a single sensor fault or another kind of fault. If the problem is due to sensors, the faulty sensor is isolated and a parameter related to the fault severity is produced as well.
- A second net is used whenever the first one detected a fault not simply due to a single sensor fault. The options are then:
  1. An engine component fault is present: isolation and severity of the fault are given
  2. There is a combination of one engine component and one sensor fault. The net flags the presence of a different kind of fault
- The third net is called whenever the second one detected a fault not simply due to a single engine component fault. This net indicates where the faults are located.

The following points have to be made:

- The modular system finds it difficult to deal with combined faults.
- The system shows a limited multi-fault capability: only one engine component and one sensor can be simultaneously faulty. If a more complex engine were analysed with a similar system, the large dimensionality would prevent application of the method.
- The measurement biases dealt with are relatively large in magnitude (2-8%). Smaller biases cannot be detected and isolated, even though their effect on the diagnostic accuracy can be very significant.
- No claim is made as to ability of quantification of the faults.

The results obtained suggest that modularity is necessary but a NN-based system designed to deal with more realistic fault situations is likely to need a larger set of nets.

Two other applications are described:

- A backpropagation net is used to try to diagnose which compressor stage is faulty. The technique performs the same task carried out by stage-stacking techniques (Mathioudakis and Stamatis, 1994). The outcome of the study is that whereas faults in the front and rear stages are correctly identified, those in the middle ones are hard to cope with. The overall performance of the NN-based method is therefore unsatisfactory.
- A backpropagation net is used to identify burner faults. Input quantities are readings of 16 thermocouples plus the norm of the temperature input vector, output quantities are 8 parameters, each one expressing faults in the corresponding burner, plus a

severity index. The classification performance is 100%. This shows how suitable NNs are to detect and isolate faulty burners.

3.3.6 The Rolls-Royce approach

Rolls-Royce have carried out an exploratory work to assess the NN's capability to perform gas turbine diagnostics (Nayer, 1994). A number of nets have been trained and tested with single and multiple fault cases. In the sequel a brief analysis of the study is reported with no details due to the proprietary nature of the information.

The relevant features of the work, together with some information as to the results obtained, are as follows:

- a 3-spool aero-engine (Trent family) has been analysed
- training and test data have been obtained by simulation
- data relative to 7 different power levels have been used (from low to high power)
- rather large data sets have been used to allow coverage of a wide range of faults and operating points
- 16 measurements were used for setting the operating point and monitoring the engine's health. This is the typical instrumentation available for production pass off test
- 19 performance parameters were used to quantify the engine component faults
- 5 different NN-based approaches have been considered
- all nets have been trained with standard backpropagation
- single hidden layer nets were used
- 2 different data sets were used:
  1. nets were first trained with single engine component faults (max deterioration 2%) and single sensor faults. Tests were done by using multiple faults as well to ascertain the nets' capability to generalise from single to multiple fault cases (up to 4 measurement biases and 4 component faults)
  2. two performance parameters were supposed to be simultaneously faulty. Biases were at most 2 out 16 measurements, with a level of 10% shift
- one of the 5 approaches simply tried to estimate 35 parameters (19 performance parameters + 16 measurement biases) according to the classic way KF-based techniques work
- 3 approaches actually neglected measurement errors
- an attempt has been done to separately carry out measurement validation before the calculation of the performance parameters

The diagnostic accuracy achieved is rather poor:

- a large "smearing" effect (see section 2.2.1) is present
- the implanted biases are often too large for their detection to represent a real diagnostic challenge (large biases are usually easier to spot)
- measurement noise is neglected
- errors are particularly large when multiple faults are present

- a limited fault detection capability is accomplished due to the choice of a limited number of fault combinations, that have been suggested by experience and, ultimately, "gut feel" for the nature of the problem.

The following observations have to be made to evaluate the performance of the proposed NN-based diagnostic system:

- no attempt has been done to test learning algorithms different from the standard backpropagation. The use of large data sets calls for reduction of the training time and the complexity of the relationships to be mapped requires algorithms able to reach low values of the cost function as effectively as possible
- as far as flexibility is concerned, NNs cannot be compared with model-based methods. If a neural net is trained with data covering a range of faults expressed by performance parameters variations of 2%, it is unrealistic to assume the net to be able to provide accurate results when presented with 3% level faults. Similarly, if a net is trained with single faults data, its accuracy on multiple faults data is likely to be rather poor. As a matter of fact, the extrapolation capability of NNs is limited and the design of a NN-based system has to take this into account
- the Auto-Associative neural network used to validate measurements has a very limited capability of isolation of the faulty sensors. This is due to unsuitable design in term of number of hidden layers and neurons.

Analysis of the performance of the RR NN-based diagnostic systems leads to the following conclusions:

- if NNs are to be used for difficult diagnostic tasks, modularity is the key point to achieve accurate results. Separation of the measurement validation from the estimation of the performance parameters seems to be a good choice
- modularity is necessary to avoid forcing the nets to extrapolate in domains that are significantly different from those used for training. As NNs can generalise well when test and training sets are similar, whenever a complex, real world problem has to be tackled, better results are likely to be attained by application of a "divide and conquer" approach. The problem, subdivided into a large number of small (and then simpler) subproblems, may become manageable by means of NNs
- if a highly modular approach is pursued, attention has to be paid to the computing power required (time and memory) for setting up the system. In this respect, use of quick and effective training algorithms is required
- if measurement validation has to be done by NNs, a technique has to be established, which is both theoretically sound and computationally feasible
- it is very difficult to obtain by NNs an estimation of the error affecting the results. Unlike KF-based methods, for most of the NNs it is not possible to get information as to the level of confidence of the results
- unlike knowledge-based systems, NNs cannot provide an explanation of the reason why a certain result has been produced. NNs can be regarded as "black box" techniques.

In conclusion, modularity and, in general, awareness of the inherent limitations of NNs seem to be key points for their successful application to such a complex problem as gas turbine diagnostics.

3.4 Sensor Failure Detection, Isolation and Accommodation using NNs

A common problem in many areas of engineering is the presence of biased measurements. This is especially true in systems where hardware redundancy is not available and the effects of a measurement error can be large and unpredictable. In this case a sort of analytical redundancy has to be used in order to continuously monitor the condition of the instrumentation set and thereby of the physical process. Problems of this kind are encountered for example in flight control systems and gas turbine engine diagnostics. Thus, a technique is searched for, which is able to find out:

1. whether the instrumentation set as a whole is affected by faults (***Detection***)
2. which sensor(s) is(are) affected by the fault (***Isolation*** or ***Identification***)
3. the amount of the measurement error, necessary to recover the actual measurement values (***Accommodation***).

It should be noted that various conventional estimation techniques (e.g. the Kalman filter, MME estimator) could be used to both estimate the system's state and perform Sensor Failure Detection, Isolation and Accommodation (SFDIA). Nonetheless, the inherent limitations of those techniques suggest that a different approach could be pursued, which was based on the use of a separate module able to perform SFDIA. After validation and possible correction of the data, a model-based technique could be used.

A main distinction can be made between:

- SFDIA for time varying processes. As far as gas turbines are concerned, this type of SFDIA method can be applied to:
  - diagnostics of transient data: a dynamic model is usually available
  - diagnostics of steady state data's trends: poor statistical information is available as to the trends of engine component faults as well as sensor faults. Therefore, no dynamic model is available.
- SFDIA for time constant processes. A typical case is the analysis of a single set of steady state measurement data coming from test bed.

In the next two sections both applications are reviewed. Suggestions for modifications to the established methods are proposed as well.

3.4.1 SFDIA for time varying processes

A broad distinction among SFDIA techniques can be done according to the kind of algorithm used:

- model-based SFDIA. The typical method is obviously the linear or non-linear Kalman filter (sections 2.2 and 2.7). Another interesting and effective tool is certainly the observer with eigenstructure assignment (section 2.5). An MME estimator could be

used for SFDIA purpose as well (section 2.8). These techniques need a measurement equation and, apart from the MME (to a certain extent), a dynamic model.
- AI SFDIA. Both expert systems and neural networks can be used to correct biased measurements. In this case diagnostics can be provided even with no modelling.

A detailed comparison has been made between the two approaches by Napolitano et al. (1996).
Sensor failures can be classified as follows:
- hard-over failures, catastrophic but easy to detect
- soft failures, difficult to detect and, if uncompensated, potentially catastrophic.

The former type of sensor failure can be assumed to be detectable by means of other techniques, while the latter has to be dealt with by means of analytical redundancy and its correction is the objective of SFDIA.

The following issues have to be considered when assessing the performance of an SFDIA technique for soft failures:
- types of failures: small bias, slow drifts, parabolic-like errors are possible
- observability of the effect of the failures from the available measurements
- amount of time required to detect a sensor failure
- uniqueness of the failure and degree of distinguishability from other types of sensor faults.

Conventional model-based techniques used for SFDIA (e.g. the Kalman filter) are affected by lack of robustness as described below:
- the robustness to non-linearities is usually low. This issue has been detailed in section 2.2
- the robustness to noise may be low. A low signal-to-noise ratio is likely to produce inaccurate results
- the robustness to modelling errors is usually low.

Moreover, these techniques show high sensitivity to:
- bad measurement and/or intermittent failure (outliers, data gaps, temporary loss of signals)
- use of reduced-order filter, because of constraints on the available computational power.

This means that a high rate of false alarms is likely and so the reliability of the diagnostics is strongly reduced.

As far as gas turbine diagnostics is concerned, the issue about modelling errors is particularly serious. As a matter of fact, unless an unsteady state fault diagnosis is chosen, the only available equation is the vector measurement model. No model is given to describe the fault dynamics and hence modelling errors are very likely to occur, even neglecting the actual problems of achieving optimality for a non-linear problem.

AI techniques can actually represent a suitable tool for SFDIA. Expert system techniques for SFDIA of gas turbine engines have already been considered in section 2.4. As far as

NNs for SFDIA are concerned, a NN structure can be used as an on-line learning state estimator for replicating sensor data.
The problems likely to be encountered with this approach are:

- a proper balance between on-line and off-line learning has to be found
- the time necessary to reach an acceptable learning level, and therefore a good detection capability, is important. This is obviously dependent on the required accuracy and the kind of training algorithm used

A well tested structure for SFDIA is displayed in fig. 3.4.

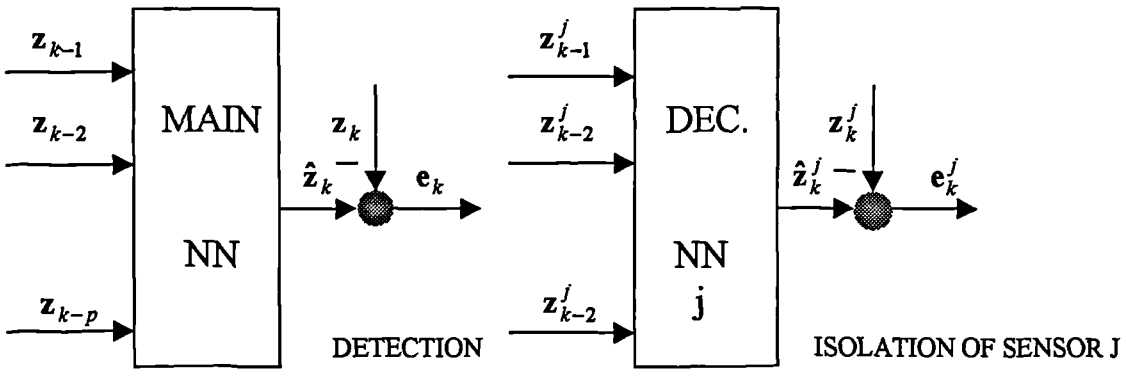


Fig. 3.4: detection and isolation of sensor faults

The structure of the SFDIA system is made of a main NN (MNN) and M decentralised NNs (DNN), where M is the number of available measurements. The main one is fed with the last p (M,1) measurement vectors (from $\mathbf{z}_{k-1}$ to $\mathbf{z}_{k-p}$) and produces a one step prediction of the measurement vector. Comparison with the set of measured quantities enables detection of possible sensor faults. The j-th decentralised NN is fed with the same data, apart from the j-th measurement. The superscript (j) in fig. 3.4 means that the j-th component of the measurement vectors has been removed. Basically the j-th DNN produces a one step prediction of the j-th measurement by using the other $M-1$ measurements. Once a sensor bias has been detected by the MNN, the bank of M DNN performs sensor fault identification. Eventually accommodation is made by substituting the output of the j-th DNN (e.g. measurement j) for the measured quantity. It is noted that as long as no fault is detected, both the MNN and the set of DNN are trained on-line. When the fault is identified, the structure of the j-th DNN is frozen, i.e. the weights are not updated anymore, and the j-th measurement is actually provided by the output of the j-th DNN.
The MNN and the DNNs are trained on-line with the Extended Backpropagation (EBP), a modified and enhanced version of the common backpropagation algorithm.

A comparison has been made by Napolitano et al. (1996) between the above NN structure and a similar structure where every NN has been replaced by a linear Kalman filter. Three main criteria have been used for comparison purpose:

- the ability of detecting faults
- the false alarm rate

- the estimation error.

Different types of sensor errors have been considered (small steps, slow drifts, combinations of both) and different levels of discrepancy between the KF model and the actual model providing the data. It is worth noting that the latter property, namely the robustness with respect to unmodelled effects, is very important for gas turbine SFDIA. The results of the comparison are summarised below:

- the KF-based SFDIA shows slightly better robustness to system and measurement noise when the model errors are small
- the on-line learning NN-based structure performs definitely better when model errors are large. It is able to track any dynamics, even non-linear
- the NN-based structure needs time to learn the dynamics of the system, but while the KF estimation's performance is bounded, NN estimation can only improve as on-line learning continues. This enables NNs to outperform KF
- even though the neural architectures are not recursive, the computational power they require is less than that of the KF
- the detection ability is equivalent for biases, while ramps are better detected by NNs
- both structures seem to have difficulties in detecting ramps.

The following remarks can be made about the applicability of SFDIA to gas turbine diagnostics:

- as the sensor fault dynamics is not known, the NN SFDIA seems more suitable
- the problem of the time necessary to let the nets learn the dynamics should be properly analysed. Basically tests should be made in order to assess the speed of learning of the nets related to the level of the sensor fault
- better results in terms of accuracy are probable if a different learning algorithm is used. In the study made by Napolitano, NNs are used for SFDIA of a flight control system. This implies that learning time is a primary issue especially in terms of computing time. If NNs were applied to gas turbine diagnostics of civil aero-engines, though, accuracy would be a primary requirement and a quick algorithm, even if desirable, would not be strictly necessary. This would allow using dynamic NNs that are more suitable to track time series. The following structures seem to be adequate candidates:
  a) Finite-duration Impulse Response (FIR) perceptron, trained with temporal backpropagation with adaptive time delays
  b) Backpropagation Through Time (BTT) nets
  c) pipelined modular recurrent network, acting as a non-linear predictor.

  An overview on the above-mentioned NN structures can be found in Haykin (1994)
- the effect of off-line learning can be twofold: on one hand it can improve detection performance and speed of learning if the sensor fault is contained in the training set, on the other hand it can worsen the system's performance if the fault is new
- an explicit way of estimating whether the temporal evolution of measurements is due to a sensor fault or to an engine module fault may result necessary, according to the technique developed by Junk and Lunderstaedt (1995).

A system suitable for on-wing gas turbine diagnostics could have the following modular structure:

- SFDIA is performed using NNs, possibly with the modular architecture described by Napolitano. Other learning algorithms (cited above) can be used and a more complex pipelined structure can be chosen as well.
- after accommodation of sensor failures, a common MME estimator (see section 2.8) can be applied.

The advantages of a system of this kind are summarised below:

- all the algorithms used are well suited to take non-linearity into account
- all the algorithms used are suitable for poorly modelled fault dynamics
- good robustness with respect to measurement noise should be achieved.

The relationship between the obtainable degree of accuracy and the number of measurements necessary for the nets to learn should be thoroughly investigated by means of extensive simulations.

A simpler form of SFDIA for gas turbine engines has been developed by Guo et al. (1996). A single NN made of 5 layers performs SFDIA. It has the structure commonly used to compress data: inputs are the same as outputs during training and a refined version of backpropagation (enhanced with a Genetic Algorithm optimisation technique) is used. Training is off-line. A threshold is selected for each output of the net and when the residual (i.e. the difference between the estimated and the measured quantity) is larger than the threshold the sensor fault is detected and identified. An 8-40-4-20-8 structure has been chosen. When a sensor fault is detected, its input is disconnected and replaced with the most recent estimate provided by the net itself. In this way proper SFDIA is possible.

The following remarks can be made:

- even though good results are claimed, the actual performance of a system of this kind should be assessed, especially when engine component faults are time varying
- the robustness of the technique to different sensor fault conditions should be properly tested, as off-line training does not allow for adaptation (a limited number of sensor faults can be considered).

In conclusion, NNs seem to be well suited to perform SFDIA for gas turbines, especially if in conjunction with a non-linear model-based estimation technique such as MME. Nonetheless proper and extensive testing is required. In particular the solutions proposed by Napolitano and by Guo should be further investigated. The issues about the need for distinction among sensor and engine component faults and the required number of samples to let the nets learn the fault dynamics should be analysed.

3.4.2 SFDIA for time constant processes

When a single set of steady state measurement data is the only input, NNs turn out to be a very effective tool to perform SFDIA.

NNs used for data validation are simply layered nets usually trained by backpropagation. A brief overview about NN-based validation for time constant systems is given below (Kramer, 1992; 1991).

NNs can be classified according to the kind of training: if the training algorithm uses different input and output patterns the learning is named *supervised*, if the training algorithm has to extract information just from input patterns the learning is named *unsupervised*. If input and output patterns are the same, the training is *self-supervised* and the net performs *autoassociation*.
Nets subject to self-supervised training can be used for data validation. The structure of a typical net is shown in fig. 3.5.

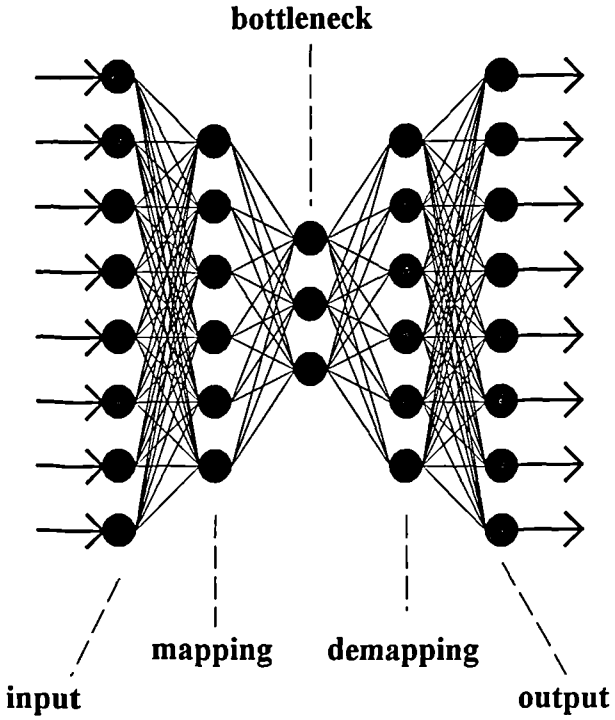


Fig. 3.5: an Auto-Associative Neural Network (AANN)

An autoassociative net has the same number of neurons in the input as in the output layer (*superficial dimensionality*) and can have one or more hidden layers. The main feature of these nets is that they can perform Principal Component Analysis if a hidden layer (the so-called *bottleneck layer*) is made of a number of neurons equal to the *intrinsic dimensionality* of the data. The intrinsic dimensionality is the number of underlying parameters that are necessary and sufficient to describe the parameters of the input space. In formula:

$$\mathbf{t} = \mathbf{g}(\mathbf{y}) \tag{3.28}$$

where:

- M is the superficial dimensionality (i.e. the number of measurements)
- p is the intrinsic dimensionality
- $\mathbf{t} \in R^M$ is a vector of the input space
- $\mathbf{g} \in R^M$ is a vector function
- $\mathbf{y} \in R^p$ is a vector of the so-called *feature space*.

If the function **g** is identified, Principal Component Analysis is achieved as any vector **t** can be expressed by its corresponding **y**. If **g** is linear the identification of the function is named ***Linear Principal Component Analysis*** (LPCA), if non-linear ***Non-linear Principal Component Analysis*** (NLPCA).
Principal Component Analysis can be done when there is some redundancy in the input space: p is less than M and therefore data compression is performed. To be of practical usage, the *mapping* to the feature space must be followed by *demapping* compressed data back to the input space. In formula:

$$\mathbf{y} = \mathbf{h}(\mathbf{t}) \tag{3.29}$$

where:

- $\mathbf{h} \in R^M$ is a vector function.

The effect of the mapping-demapping process is twofold:

- input noise is reduced
- input biases are removed.

The elimination (or at least reduction) of input data noise and biases actually takes place in the mapping, whereby the redundancy is exploited for compression purposes.
NNs can perform both mapping and demapping by training a net like the one in fig. 3.5. During training the input vectors are noisy, the output vectors are the corresponding noise-free vectors. A trained net is able to compress input data by mapping to the principal components of the feature space, which are the output values of the bottleneck layer. Thus mapping is performed through the transformation occurring from the input to the bottleneck layer, demapping (i.e. data reconstruction) through the transformation occurring from the bottleneck to the output layer. Ideally, after training output values should be noise-free.
The number of hidden layers is chosen whether LPCA or NLPCA has to be carried out. If **g** and **h** are non-linear, NLPCA is necessary and therefore 5 hidden layers have to be used, as a 3 hidden layer net is necessary for mapping and a 3 hidden layer is necessary for demapping. It is reminded that any non-linear function can be properly approximated by a 3 hidden layer net with non-linear activation functions in the hidden layer (see the universal approximation theorem in section 3.2.1). Therefore the activation function of the first and third hidden layers has to be non-linear, whereas input, bottleneck and output layers can have linear activation functions.
If **g** and **h** are linear, mapping and demapping phases are linear and therefore a single hidden layer is sufficient.

So far, NLPCA has been considered just as a tool to reduce noise. It can actually perform bias detection, isolation and accommodation as well.
Training a net as described above enables mapping of the identity function, apart from noise effects. However, if an input vector is biased, the output will be different from the input and thus an input bias is detected (fig. 3.6).

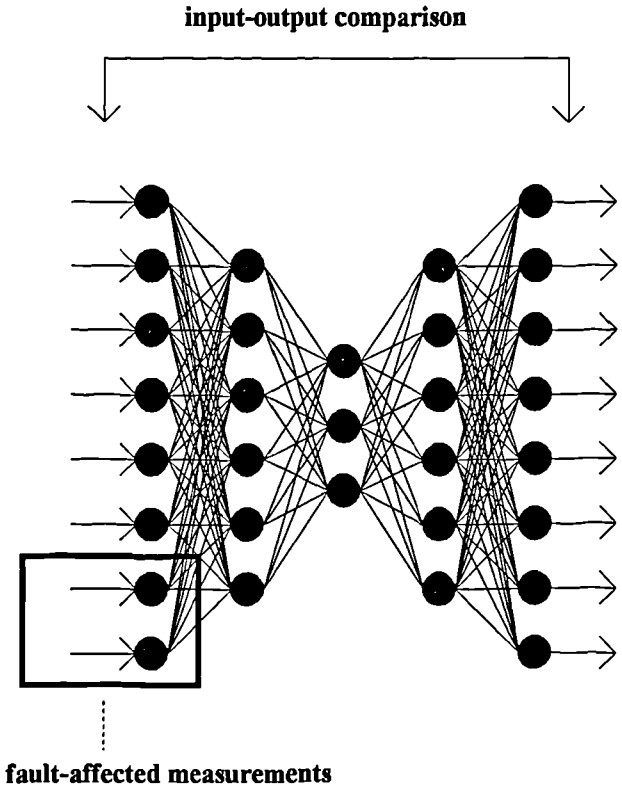


Fig 3.6: bias detection

If the RMS error (Root Mean Square) is larger that a fixed threshold detection is obtained, but statistical tests are necessary to identify the biased components of the input vector. A straightforward statistical quoting technique relying on the magnitude of the errors of the various components is simplistic and misleading. In particular the *measurement test* proposed by Kramer (1992) seems to be theoretically sound. Nonetheless, violation of the measurement test does not guarantee that the associated input element is biased (Mah, 1990).

A better approach consists in a *serial minimisation* approach. As the net has actually learnt how to map the identity function, it only outputs data different in a statistical sense from the input data when the input vector is sensibly different from training set data and hence cannot be properly mapped to the non-linear principal components expressed by the bottleneck layer activation values. If a component of the input vector is biased, restoration of a RMS consistent with the training set one can be done by substitution of the biased component for the corresponding unbiased one. The latter can be found by minimising the RMS as a function of the biased component. The input component, which after minimisation leads to a minimum RMS, allows both identification and accommodation of the biased input. Of course a serial minimisation must be applied, as the input component must be varied one at a time. If multiple bias identification and elimination is required, all possible combinations have to be considered.

Another approach to filter out both noise and biases consists in the use of a *Robust Auto-Associative Neural Net* (RAANN). In this case training is split in two separate phases:

1) a common 5 hidden layer net is trained to map the identity function. Inputs are noisy and outputs are noise-free. The corresponding training set is Y. After completion of the training, the post-bottleneck weights are saved. They will be used by the final RAANN.
2) the training set is enlarged by sequentially biasing every input pattern. If q multiple biases are supposed to be possible, then for each bias-free pattern all q combinations are considered and biased. The corresponding training set will be made of input vectors of this kind:

$$\mathbf{z} = \mathbf{y} + \delta_j \mathbf{e}_j \qquad j = 1,...,M \tag{3.30}$$

where

- M is the number of measurements
- $\mathbf{y}$ is the bias-free vector
- $\mathbf{e}_j$ is the j-th column of the identity matrix
- δ_j is the artificially imposed bias for the j-th measurement.

The bias is usually chosen as a multiple of the standard deviation:

$$\delta_j = \pm a \cdot 3 \cdot \sigma_j \tag{3.31}$$

where:

- σ_j is the standard deviation of the j-th measurement
- a is a constant larger than 1 to avoid replication of patterns.

As biases can have different values, a is given several values (usually $a = 2,3,4,....$).
The corresponding output patterns are obtained by simple compression of the bias- and noise-free inputs through the pre-bottleneck part of the net trained in the first step. They are simply the activation values of the bottleneck neurons.
In this way a comprehensive training set is created, including both biased and bias-free measurement vectors ($Z + Y$).

Whereas in the first phase a 5 layer net is trained with bias-free data, in the second phase a 3 layer net is trained with faulty data as well. The second net has the same number of inputs as the 5 layer net and a number of outputs equal to the intrinsic dimensionality of the problem, i.e. the number of bottleneck neurons of the first step net. The single hidden layer has a large number of neurons due to the high complexity of the mapping task, as in this case any biased input must be mapped onto its fault-free principal component vector. Once training of this single hidden layer net is completed, the RAANN can be built: the pre-bottleneck structure comes from the second step net, the post-bottleneck structure comes from the first step net. The $Z + Y$ training set could actually be directly used to train a 5 layer net. However the two step training procedure is convenient because in the most difficult part of the learning (filtering of both noise and biases) a single rather than 3 hidden layer net has to be trained. The final RAANN is shown in fig. 3.7 for the case of 8 inputs and an intrinsic dimensionality of 3. It should be noted that the number of neurons in the mapping layer is larger than the number of neurons in the demapping layer due to the higher complexity of the training task.

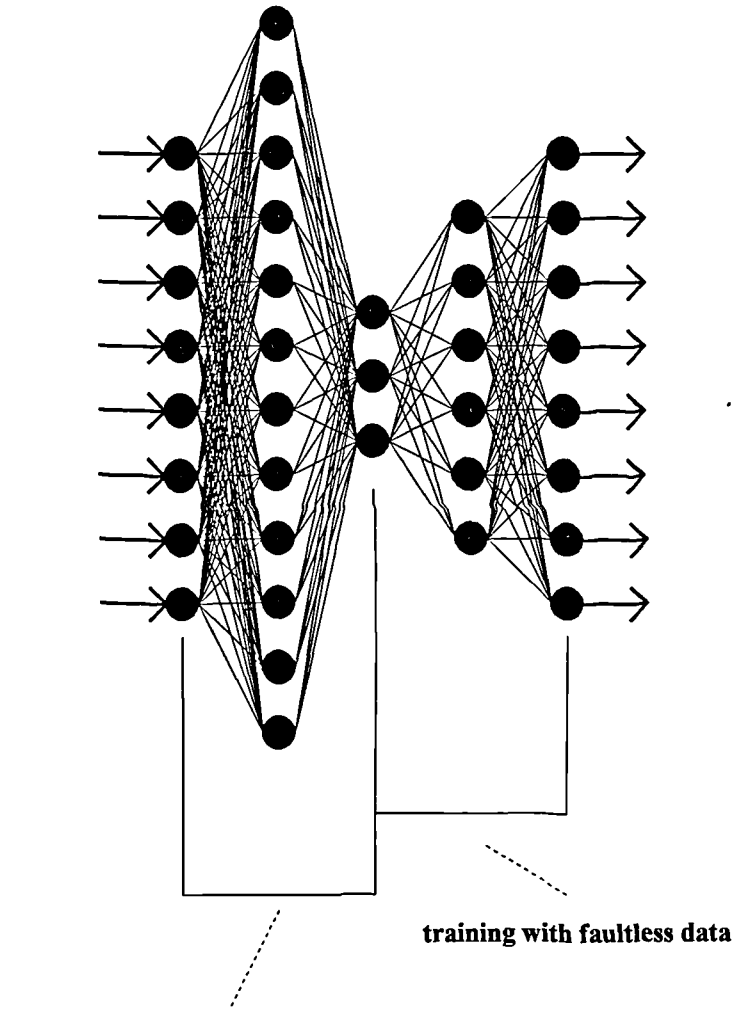


Fig. 3.7: a RAANN

Thus, when a noisy and biased input vector is given, the pre-bottleneck part of the net performs data compression (and therefore noise reduction and bias elimination) and the post-bottleneck part demaps the principal components back to the input space. The output vector should be the unbiased version of the input one and noise should be reduced as well. Therefore once training is completed RAANN performs SFDIA in a single step.

In conclusion, if bias-free input data are used for training a sequential minimisation technique has to be employed, if biased input data are used a RAANN has to be employed. In both cases complete SFDIA should be attained.

When applied to engine measurements affected by component deterioration, the following points must be made:

- NLPCA is necessary, as the relationships are non-linear
- data validation is possible only when the number of varying parameters is less than the number of measurements, otherwise no redundancy can be exploited

- mapping and demapping are quite complex non-linear functions and hence a large number of hidden neurons have to be used in the mapping and demapping layers.

In the following section a brief review of the work published on AANNs applied to SFDIA is given. Sections 3.5 and 3.6 describe the author's work on NN-based gas turbine diagnostics. In particular section 3.6 analyses measurement validation through AANNs in a greater detail.

3.4.2.1 SFDIA through AANNs trained with steady state data

Neural architectures similar to the ones described in section 3.4.2 have already been used to detect, isolate and accommodate sensor faults in gas turbines. Two major works are briefly reviewed below. In this respect, it is important to highlight that in these cases:

- the AANNs have been trained with a large number of steady state data related to different operating points so as to cover a sufficiently wide area in the engine's operational envelope
- the AANNs have been tested with time-varying faults. It is therefore assumed that the time history of the various quantities is simply a series of equilibrium states. This assumption is obviously an approximation which seems not to affect the accuracy of the diagnostic results significantly
- the NN-based sensor validation scheme is embedded into a gas turbine engine control system in order to replace biased measurements that are used in the control loop.

Guo et al. (1996), Mattern et al. (1997; 1998) have developed a sensor validation system with the following features:

- 7 measurements are to be validated
- 4 parameters are used to set the operating condition of the engine (power lever angle, ambient temperature, Mach number and altitude)
- an AANN is used whose structure is 7-10-4-10-7. The number of bottleneck neurons equals the number of environment and power setting parameters defining the measurements
- learning is through batch backpropagation. Kramer's method has been applied apart from the following points:
  1. Training is made of two steps: the first training with bias-free data defines the values of the weights that are the initial values for the second training carried out with biased data. The same net is used in the two training phases.
  2. During the second training, learning was fostered by preventing the modification of the first layer weights connected to the biased input.
- Two different training approaches are proposed for the second phase: the standard one and one based on threshold logic. The latter is likely to provide more accurate results, provided training itself is not too difficult.
- A simple threshold logic is used to detect and isolate sensor faults.
- The overall SFDIA performance of the AANN is good.
- The second step training is difficult and lengthy.

Moller et al. (1998) have applied AANNs to turboshaft engine SFDIA. The following remarks can be done:

- 7 measurements are monitored, which are defined by 4 parameters
- the number of underlying parameters depends on parameters used in the control system as well
- a two step training similar to the one proposed by Mattern et al. (1998) is used
- random measurement noise has been neglected
- the biases range from 10 to 100% of the unbiased value and as such are to be considered very large
- the steady state tracking is very accurate
- the transient tracking is not accurate and that represents a problem when the signal is a relevant input for the control system
- multiple biases can be coped with provided sufficient redundancy is available.

In conclusion, the following observations can be done:

- the AANN-based SFDIA methods produced accurate results, even when used to track transient behaviour. In particular the nets were able to cover a wide area of the operational envelope
- no attempt has been made yet to account for engine faults
- long and difficult training has been encountered
- the biases the nets have been trained and tested with were rather large in magnitude.

3.5 Fault diagnosis of a turbofan engine using neural networks: a quantitative approach

The present section describes a work published by the author (Zedda and Singh, 1998) on gas turbine diagnostics based on neural networks. The main purpose of the work is to test NNs as applied to the whole gas turbine diagnostic problem.

The review proposed in chapter 2 highlights the various problems encountered when real world effects, that are mainly due to measurement uncertainties, are accounted for. The successful works published on NN-based diagnostics, along with consideration of the NN inherent features (see section 3.3), suggest that a thorough test is necessary to find out to which extent NNs can represent a valid diagnostic tool.
In particular, NNs can be used for a wide range of tasks because of their flexibility. However, it is of the utmost importance to establish which tasks NNs can perform best.
Thus, the work presents a massive use of neural structures in order to test their suitability for gas turbine diagnostics.
An NN-based diagnostic system is developed, which is characterised by the following features:

1. it quantifies the amount of deterioration affecting the various engine components as variations of performance parameters
2. it detects multiple faults
3. it carries out the fault diagnosis by using the limited instrumentation set present on wing

4. it works with noisy measurements
5. it performs single Sensor Failure Detection and Isolation (SFDI) and in case of sensor fault it provides the diagnostic answer by using only the fault-free measurements.

The importance of the above-mentioned properties is highlighted and expanded below.

1. The most interesting and somewhat new feature of the developed system is the capability of quantifying the fault. So far, the described neural systems performed a classification. During training the nets were taught how to recognise a small number of faults so that when provided with an input pattern, they classified it (e.g. fouled compressor, eroded turbine, and so on). No or little information was obtained about the amount of deterioration affecting the engine components, i.e. the fault severity. Moreover, the number of detectable faults was small, because every fault had to be represented by an output neuron. If a large set of faults had to be detected, the consequent large number of output (and thereby hidden) neurons would have increased the nets' size too much. The quantitative approach that has been pursued partially overcomes this problem.
2. Another great advantage of the developed system is the capability of detecting multiple faults. Real engines are likely to be affected by faults influencing more than a single performance parameter simultaneously. For instance, fouling will modify both efficiency and flow capacity of the faulty component. Moreover, occurrence of faults affecting more than a single components (e.g. two) cannot be neglected. Diagnostic systems able to only detect single faults are of little use. Actually one of the most useful capabilities of GPA is multiple fault detection and the proposed technique is the only one able to provide multiple fault capability using NNs.
3. The small number of sensors usually present on board of the engine for on wing diagnostics seriously limits the applicability of GPA. The ability of NNs to work even with incomplete input data has been properly exploited.
4. Measurement noise has to be taken into account especially because of the large value of the noise to signal ratio: the measurement distortion due to the noise is often of the same order of magnitude as the variation affecting the measurements due to a fault of the engine component. NN robustness with respect to input noise has proved to be effective.
5. The probability of a sensor fault is high and this can substantially reduce the reliability of the diagnostic system. It is reminded that a desirable property of a diagnostic system is that its reliability be greater than the reliability of the physical process to monitor. If the system can detect and identify sensor faults, diagnostics should be possible with the remaining fault-free sensors. This is accomplished by the developed system, provided that only one sensor fault is present.

The diagnostic system is developed for a low by-pass ratio turbofan engine, the Garrett TFE 1042-70. This engine is a two-spool turbofan with afterburning. The by-pass ratio is 0.4. It incorporates a 3-stage fan and a 5-stage axi-centrifugal high-pressure compressor. It has variable geometry at inlet of the compressor. The combustor is a through flow annular type. Single stage axial turbines drive the fan and compressor rotors on concentric co-rotating shafts. The afterburner is fully modulating from minimum to maximum and uses fan discharge air within a cooling liner to maintain low

outer metal temperature. A single power takeoff is used for starting power input and power output to an aircraft furnished accessory drive.
The complexity of the engine ensures a proper test of the actual diagnostic capability of NNs. The measurements that will be present on board of the aircraft powered by this engine (Indigenous Defensive Fighter, made in Taiwan) are 7 ($M = 7$, where M is the number of measurements): W_2 (engine inlet air flow), P_3 (HP compressor delivery total pressure), W_{fmb} (main burner fuel flow), N_1 (LP spool speed), N_2 (HP spool speed), T_{tet} (LP turbine exit total temperature), P_{16} (fan by-pass duct total pressure). The diagnostic system should use only these 7 measurements.
For a twin-spool turbofan like the TFE 1042, a number of 8 parameters are necessary to assess the components' health. Therefore, a requirement for the diagnostic method would be the calculation of efficiencies and flow capacities of fan, HP compressor, LP and HP turbines ($\eta_{FAN}, \Gamma_{FAN}, \eta_{HPC}, \Gamma_{HPC}, \eta_{HPT}, \Gamma_{HPT}, \eta_{LPT}, \Gamma_{LPT}$). On the other hand, as the suitability of NNs for sensor fault detection and isolation is to be tested as well, it is assumed that one out of four engine components is not affected by faults. In this way, once the fault-free engine component is isolated, the diagnostic system has to calculate the remaining 6 performance parameters. This choice allows the exploitation of analytical redundancy for single sensor fault diagnosis, as 7 measurements are available.
The database necessary to train the networks of the diagnostic system is obtained with a simulation program able to reproduce the steady-state aero-thermodynamic behaviour of the engine. TURBOMATCH, a well-tested Cranfield simulation software has been used. The program, used in synthesis mode, calculates the values of the 7 measurable quantities starting from the imposed values of the 8 performance parameters by using non-linear aero-thermodynamic equations and real component characteristics.
Two different kinds of fault are considered:

- small deteriorations affecting a number of parameters ranging from 1 to 6 (category A)
- large deteriorations affecting only 1 or 2 parameters (category B).

Both dependent and independent parameters can be expressed as deltas:

$$\Delta Z = \frac{Z - Z_{\text{baseline}}}{Z_{\text{baseline}}} \cdot 100 \tag{3.32}$$

where $Z_{baseline}$ is the value of the established baseline condition and Z is the measured or calculated value for measurements and performance parameters respectively. Fig. 3.8 shows the effect of measurement uncertainty on the delta value. Aim of the diagnostic system is to somehow eliminate the non-repeatability and bias effects to calculate the performance drop due to component degradation.
Since the system should be able to detect generic multiple faults, it is necessary to use a very large database. This is made of several couples of measurement-performance parameter vectors; every couple will be called *example*. The category A fault library is made of every possible 3-component fault combination represented by 6 out of the 8 performance parameters taking on integer values ranging between 0 and 2. The category B fault library is made of every possible dual fault combination with independent

parameters ranging from 0 to 5. Moreover, flow capacities deltas can be positive as well as negative.
The fault library so results in 14838 examples (13500 from category A and 1338 from category B). The operating condition is sea level static, at full power.
Two remarks have to be done about the two categories:

- even GPA finds it difficult to accurately calculate multiple faults represented by such a large number of performance parameters different from zero as in cat. A. If the GPA is linear, then the usual problems of inaccuracy are likely to occur, even though the linearisation is somewhat acceptable because of the low level of deterioration. If the GPA is non-linear, problems of convergence might occur due to the high dimensionality of the equations. Escher (1995) pointed out the influence of the choice of the measurement set on the convergence of non-linear GPA
- category B faults are at the same time unlikely and dangerous. Their detection is very critical for both economic and safety reasons and is better done by means of a non-linear technique. Even conventional estimation techniques find it difficult to detect these large deterioration faults and therefore they represent a benchmark to test the usefulness and efficacy of the diagnostics (Doel, 1994b).

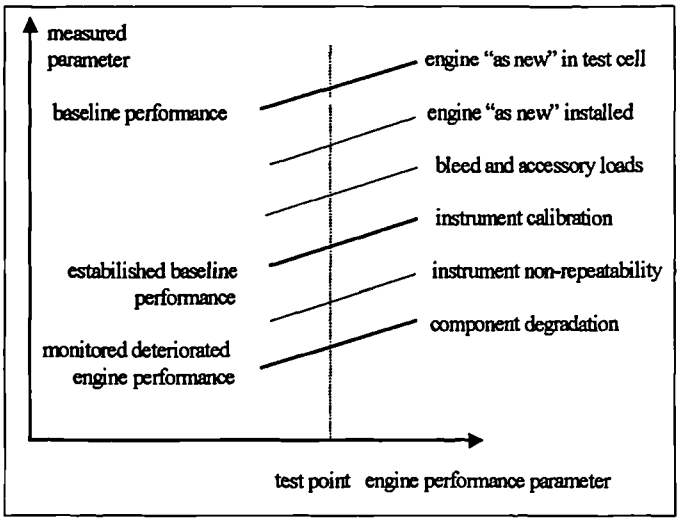


Fig. 3.8: effect of measurement uncertainty

measurement:	$3 \cdot \sigma$
ΔW2	0.5
ΔN1	0.05
ΔN2	0.05
ΔPt3	0.1
ΔWfmb	0.05
ΔTet	0.5
ΔPt16	0.1

Table 3.1: sensor non-repeatabilities

As the training set measurement vectors are produced with a simulation program, it is necessary to corrupt them by superimposition of noise to reproduce the non-repeatability effects. Of course every measurement has its own range of non-repeatability, as shown in table 3.1. The quantities shown are three standard deviations of the corresponding delta. The noise is uniformly distributed around the mean.

From an analytical viewpoint the system must perform a multivariate regression to calculate 8 performance parameters from 7 noisy measurements. The main objective is so an acceptable approximation of the function:

$$F: X \rightarrow Y \qquad (3.33)$$

where $X \subset R^7$ is the set of measurement vectors and $Y \subset R^8$ is the set of performance parameter vectors, i.e. the set of considered faults. Whenever one engine component assumed to be fault-free is isolated, the system should be able to concentrate on the remaining components and hence the above function is modified as $Y \subset R^6$.
"Acceptable" means comparable with the performance of the current analytical techniques and compatible with the diagnostics' accuracy requirements. Moreover, the system has to reject a biased measurement. The first goal can in theory be attained with various kinds of neural architecture: feedforward backpropagation, recurrent backpropagation and radial basis function networks seem to be suitable.
The presence of a high level of noise in the measurements and the objective to detect all possible faults represented by a set of more than 14000 examples make the problem of the function approximation (3.33) very difficult, also because the required accuracy is high. A ***modular*** neural system using noisy data is developed which simplifies the regression task by dividing the problem in a number of smaller (and hence simpler) problems. Fig. 3.9 displays the modular structure of the diagnostic system. In the sequel the various modules are described.

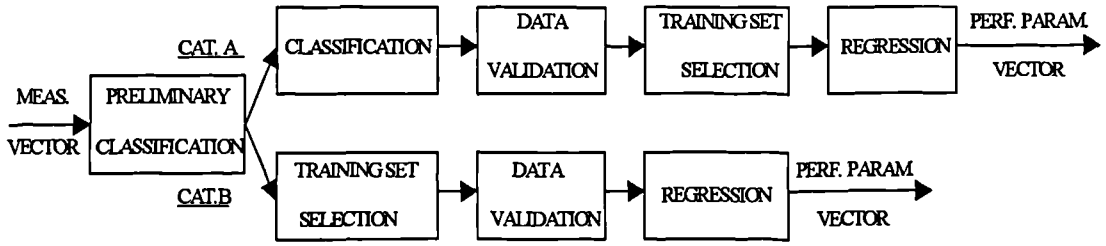


Fig. 3.9: layout of the diagnostic system

3.5.1 Preliminary classification

This module classifies the input measurement vectors according to the fault category (A or B). This first classification overcomes the problem of the low diagnostic accuracy usually obtained when the two categories are mixed. Moreover, since the characteristics of the two fault categories are different, separate diagnostics can be developed.

A feedforward backpropagation network, made of 7 input, 30 hidden and 2 output neurons performs the classification. If the network output is near (1;0) then the input measurement vector is recognised as a member of category A, if the output is (0;1) as a member of category B. The training set is made of 1000 examples from category A and 1000 from category B. The test set is made of 600 examples, a half from A and a half from B. A 3000 epoch training is enough to get good generalisation.

This approach to the classification problem is useful because it also suggests the difficulty of the classification task: if the net output is very close to (1;0) (or to (0;1)), the classification is likely to be correct, if the net output is for example (0,6;0,4) (or (0,4;0,6)), the classification may be incorrect. However, the classification performance of this module is excellent, also because the different features of the 2 categories of examples make the task easy.

3.5.1.1. Category A fault diagnosis

The more common case of category A faults is considered first and the various diagnostic steps explained.

3.5.1.1.1 Classification

This module performs a classification based on the value of the performance parameters. The module is made of 8 nets trained with the backpropagation algorithm.
Each of the 4 nets for the flow capacity has 7 input, 25 hidden and 3 output neurons and calculates the sign of a flow capacity delta. If the flow capacity delta is positive, the net should produce an output near (1;0;0), if negative near (0;0;1), if zero near (0;1;0).
Each of 4 nets for the efficiency has 7 input, 20 hidden and 2 output neurons and estimates whether the considered efficiency delta is positive or zero. If the efficiency delta is positive, the net should produce an output near (1;0), otherwise near (0;1). As stated before, this approach also provides a measure of the classification's confidence level.
The training sets are made of 1500 examples, as well as the test sets. The generalisation properties of the nets are very good and convergence is reached after few epochs: 500 are sufficient.
These 8 nets show that classification with the available measurement set is easier for fan and HP turbine (100%) than for HP compressor and LP turbine (98% and 96%).
The overall classification performance of this module of the system turns out to be very high also because one engine component is not fault-affected and the probability of misclassification of both its performance parameters is very small. Moreover, the misclassifications on flow capacity are such that a positive or negative delta can be classified as zero (or vice versa) but a positive delta is never classified as negative (or vice versa).
The classification creates 32 classes, each one made of examples with performance parameter vectors having in general 6 non-zero deltas and flow capacities with the same sign. Every class contains the examples with zero capacity sign as well.

3.5.1.1.2 Data validation

This module is able to detect a bias present in the measurement set. Once a measurement is recognised as faulty, its value is discarded and the rest of the diagnostics is carried out with the remaining 6 measurements. Therefore, the module actually performs single Sensor Failure Detection and Identification (SFDI).

A single Robust Auto-Associative Neural Network (RAANN) trained for every class with the delta-bar-delta algorithm is used. The net is trained in two separate steps.

During the first step the structure is made of 7-15-6-15-7 neurons and the training and test sets are both made of 100 bias-free examples extracted from the considered class. In each example the input vector is noisy whereas the output is the corresponding noise-free vector. It is possible to reach a test set RMS as low as 0,08 after about 2000 epochs. Fig. 3.10 shows the typical learning curve (Root Mean Square error vs. number of iterations). When the training is stopped, only the weights downstream of the bottleneck layer are saved.

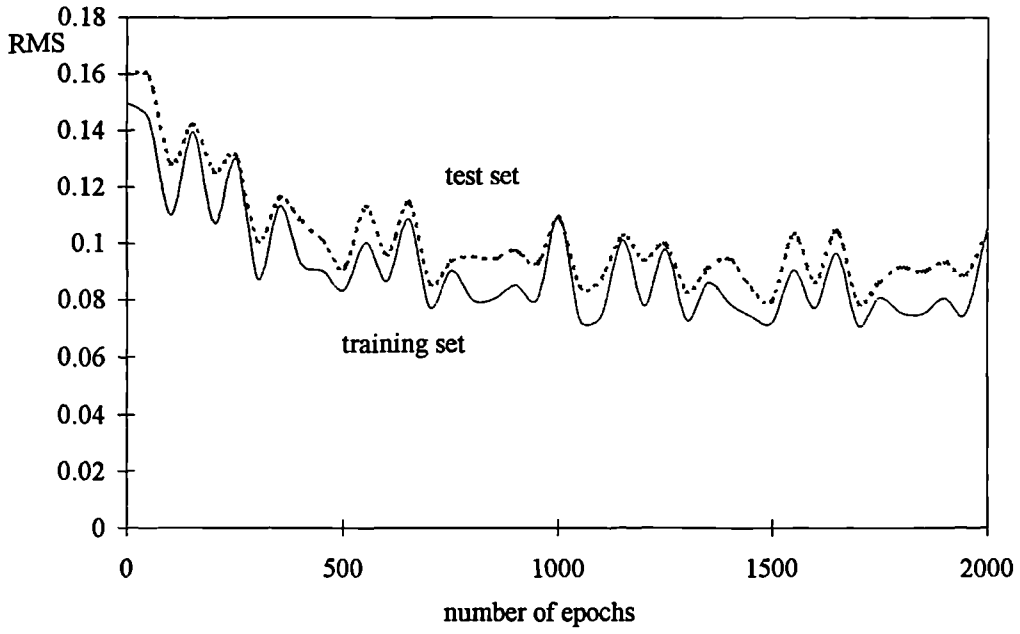


Fig. 3.10: typical first step training

The second step is the training of a 4 layer net. The training set includes noisy bias-free and noisy biased measurement vectors as input. The output vectors of the training and test set are the 6 principal components obtained by inputting bias- and noise-free measurement vectors in the above-mentioned 5 layer net and picking the activation values of the bottleneck layer neurons. It is worthwhile to highlight the novelty of the second step training proposed. In order to ease the training, the demapping part of the net is removed. If the demapping part is not removed but simply not modified by the training algorithm (weights are frozen), training actually turns out to be more difficult.

In this way a training and a test set of 1500 examples each is created. These sets are used to train and test a net with 2 hidden layers and made of 7-25-10-6 neurons. A double rather than a single hidden layer net is used as better training has been observed. Fig. 3.11 and fig. 3.12 show the corresponding learning curves. The delta-bar-delta algorithm is again used to train the net and typically a long training of about 10000 epochs is necessary to get a test set RMS of about 0,30 on the principal components.

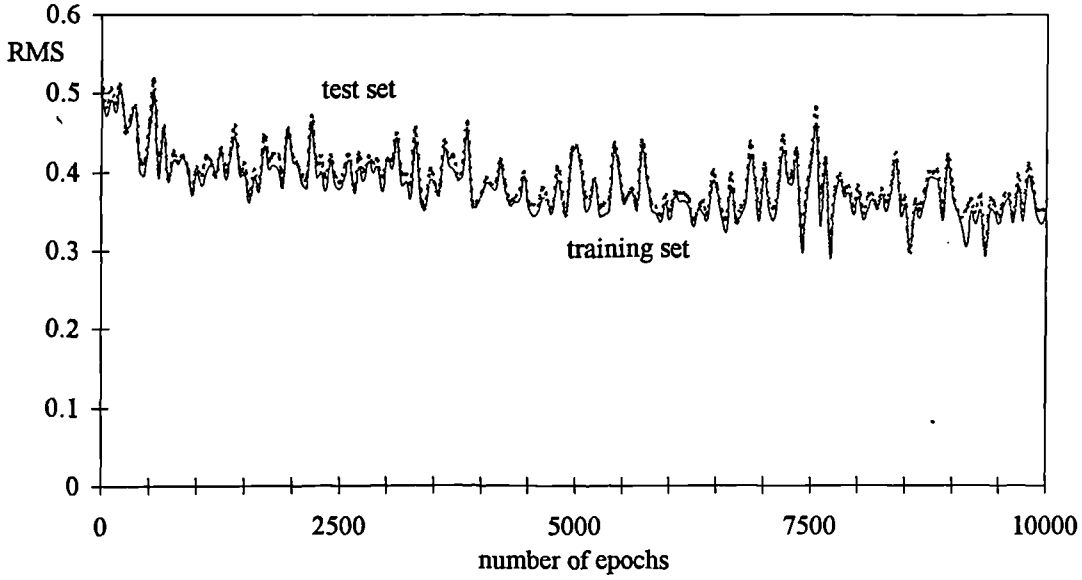


Fig. 3.11: a typical 2 hidden layer net training

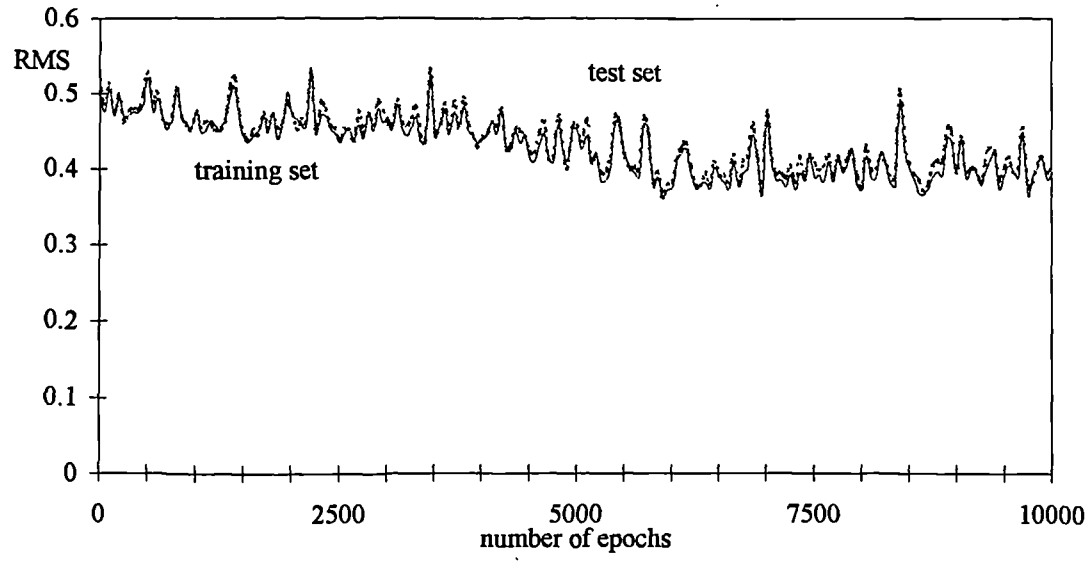


Fig. 3.12: a typical 1 hidden layer net training

Due to the features of the RAANN, proper Sensor Failure Detection, Identification and Accommodation should be possible. On the other hand, replacement of a faulty measurement has not been embedded into the diagnostic system because of the following reasons:

- the accuracy of accommodation achieved is not high. On one hand the relatively low measurement redundancy makes the second step training difficult. Trials with measurement patterns produced with variation of just 4 performance parameters (hence 4 bottleneck neurons) have shown that accommodation is feasible. On the other hand the training algorithm might not be the most suitable for the learning problem at hand. In particular, training of the 2 hidden layer net with the delta-bar-delta has shown to be difficult.
- the subsequent diagnostics is not strongly affected by the loss of information due to the rejection of a measurement. This occurs because NNs can robustly provide an estimate of the performance parameters even by using a reduced instrumentation set made of just 6 measurements.

Single SFDI is carried out by fixing two thresholds on the RMS difference between the input and the output vector. When the RMS is larger than the first threshold, then a fault of the instrumentation set is detected and the faulty sensor is isolated as corresponding to the largest error. When the RMS is between the first and the second threshold, then the sum of the errors weighted by the measurement non-repeatability is checked. If it is larger than its threshold, then the larger error to non-repeatability ratio isolates the sensor error.

In the diagnostic method data validation is made after the classifications (see fig. 3.9). From a logical viewpoint, it would be better to carry out the measurement check before any classification. However, the reasons for a choice of this sort are:

- the classification nets have shown to be very robust, being able to perform well even though a measurement is too distorted
- the data validation nets can provide good performance only if the training set is made of vectors very similar to each other. The use of these nets with the whole fault library strongly reduces their effectiveness
- the data validation module enables a sort of double check on the classification.

3.5.1.1.3 Training set selection

Since the number of vectors of any single class is still very large and the measurement noise level is high, the accuracy obtainable with a system made of 32 nets, each trained to approximate the function (3.33) for one class, is still too low. Therefore it is necessary to simplify the final regression task further. The approach is as follows: when the system is provided with a measurement vector for which the diagnosis has to be made (the so-called *diagnosis vector*), after the classification and the data validation steps a selection is done in order to extract a small number of vectors from the class database. The chosen examples should have measurement vectors somewhat similar to the diagnosis vector. In this way it is possible to minimise the final RMS.

Various kinds of selection criteria can be used, first of all the minimum distance, which draws measurement vectors from the class database starting from those having the smallest Euclidean distance.
Here another criterion is used which is able to exploit the a-priori knowledge about the different non-repeatability ranges, ensuring so a lower final RMS. Figure 3.13 shows the geometrical meaning of the search method, in the simplified case of a 2-dimensional space instead of the actual 7-dimensional measurement space (or 6-dimensional in case of sensor fault) where the selection is done.

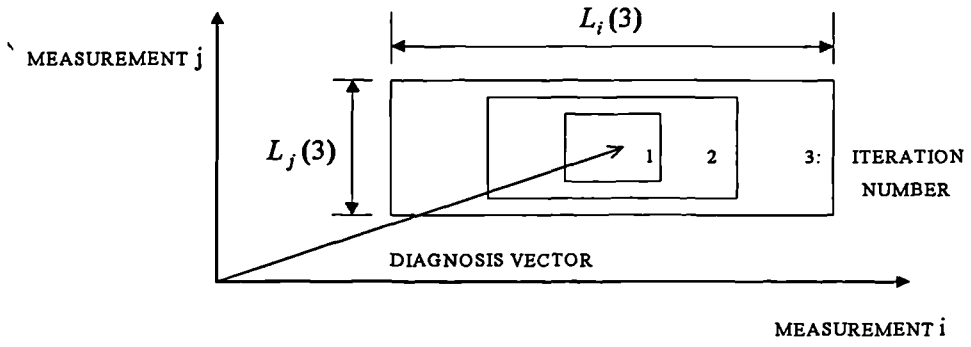


Fig. 3.13: schematic of the training set's selection algorithm

The volume containing the second extremes of the training set vectors is at first a small hypercube that expands according to the following iterative formula:

$$L_i(n) = L_i(n-1) + 2\alpha \cdot \sigma_i \tag{3.34}$$

where:

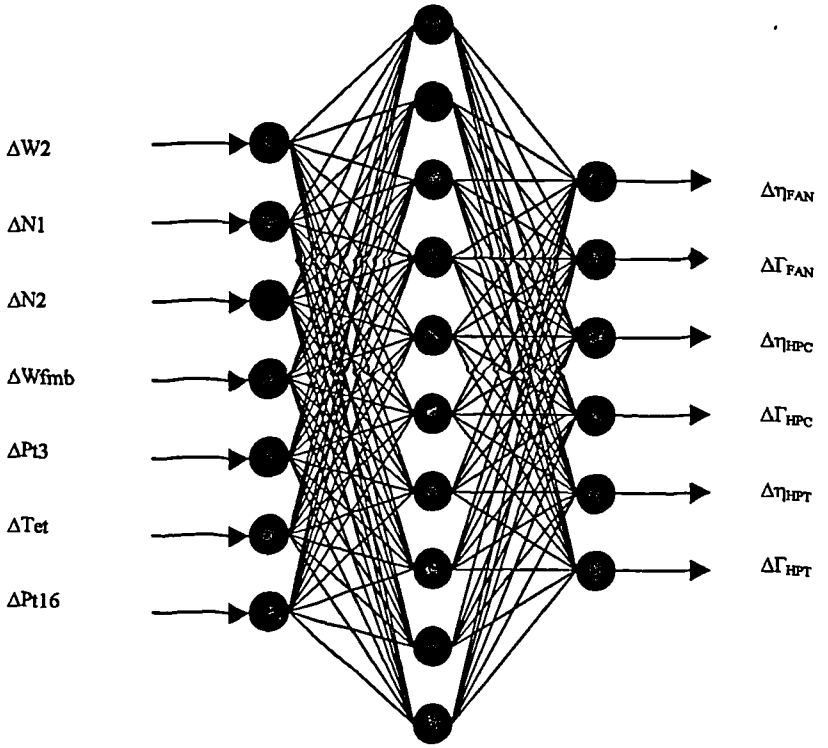
- $L_i(n)$ is the size of the selection volume along the i-th dimension at the n-th iteration
- $3 \cdot \sigma_i$ is the non-repeatability range of the i-th measurement
- α is a constant, set to 0,25.

A number of vectors is chosen and the iterative process is stopped as soon as the number of selected vectors is equal to or greater than the desired number.

3.5.1.1.4 Regression

This module performs the final approximation to calculate the 6 performance parameter deltas by using the training set selected with the previous module. The task could be worked out with different NNs. Recurrent backpropagation and radial basis function nets (Haykin, 1994) have been tested but have shown not to be suitable for the problem at hand. Training with the recurrent backpropagation has led to problems of convergence and local minima. Training with radial basis functions has not been satisfactory due to the large noise affecting the measurements and the limited capability of generalisation

shown. It should be noted that the proposed diagnostic system is made of three modules working off-line (the first three ones), for which the training is made only once, and two working on-line. For the latters, every time a diagnosis is required, a training set has to be created and hence training can be carried out. Therefore, it is particularly desirable to use a training algorithm able to get low errors very quickly. For this purpose, the delta-bar-delta training algorithm is used because it actually shows a faster convergence and the final RMS is often lower than that obtained with the standard backpropagation for the problem at hand. A suitable number of hidden neurons for the net is 10 (fig. 3.14).



Note: the LPT is not deteriorated

Fig. 3.14: regression net for cat. A

Several tests have shown that the number of the training set vectors has to be neither too small (poor generalisation) nor too large (the training set is made of vectors too different from the diagnosis vector). A suitable number for both accuracy and learning speed is about 50.

Once the training set is selected, a test set is needed to keep track of the net generalisation capabilities, since the training has to be stopped when the test set RMS starts increasing epoch by epoch, even though the training set RMS may keep on decreasing. A small number of vectors originally contained in the training set may be used, provided they are not too close to the diagnosis vector. In that case, indeed, they are useful for training. A good choice is to extract about 5 vectors from the bound of the volume containing the training set vectors' second extremes.

The way the training set is chosen is very effective, because the diagnosis vector RMS results much lower than the training set RMS, being the former in the middle of the volume where the training set vectors lie. A typical training is displayed in fig. 3.15, where the lower RMS of the diagnosis vector is shown.

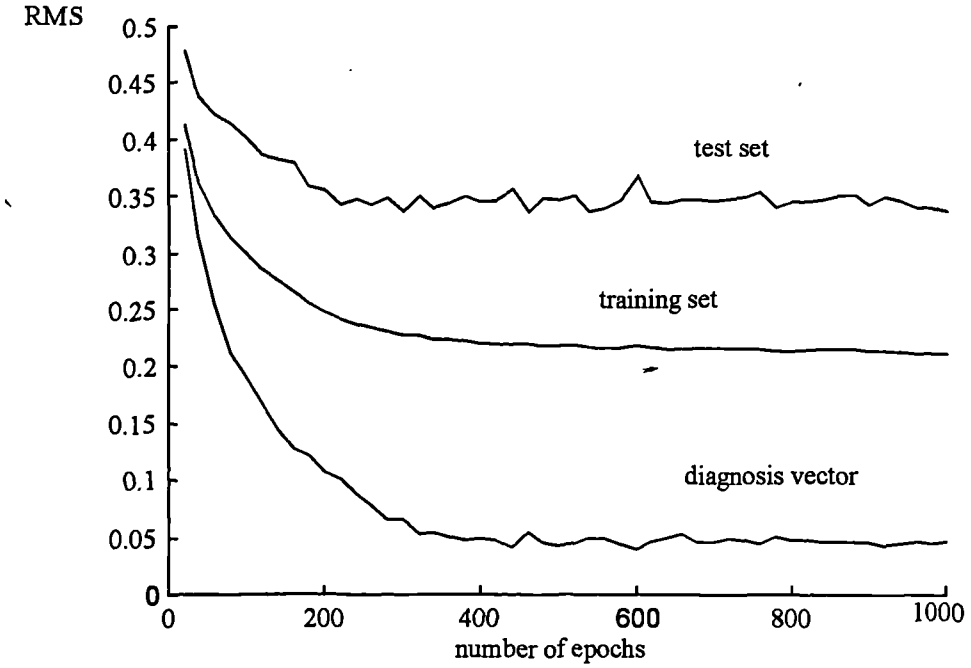


Fig. 3.15: a typical training of the regression net (cat. A)

3.5.1.2 Category B fault diagnosis

In this case diagnostics is different because:

- the category is made of only 1338 examples
- measurement vectors related to performance parameter vectors with some different sign are very different from each other, i.e. they are distant in the measurement multidimensional space.

For these reasons, a simple search method based on the minimum Euclidean distance in the space of the measurements enables to find all the examples whose performance parameter vectors have the same flow capacities signs.

After the selection of the training set, data validation is performed with the usual nets trained with backpropagation. This time 2 bottleneck neurons are used and training can even be performed on-line because convergence is very fast. The RAANN is often able to replace the faulty measurement in this case, but simple SFDI is used because the loss of information due to the rejection of a measurement does not affect the final estimation, as only 2 performance parameters have to be calculated.

The regression task can be carried out by a net trained with the delta-bar-delta algorithm and made of 7 input, 10 hidden and 2 output neurons. The training is easy and the final RMS is acceptable.

It is important to highlight that no test set is used here, as overtraining has never been encountered. In this case, therefore, training is stopped when the training set RMS curve

is sufficiently flat. The final diagnosis vector RMS is low because of the similarity with the training set vectors. Fig. 3.16 displays a typical training for the final regression step.

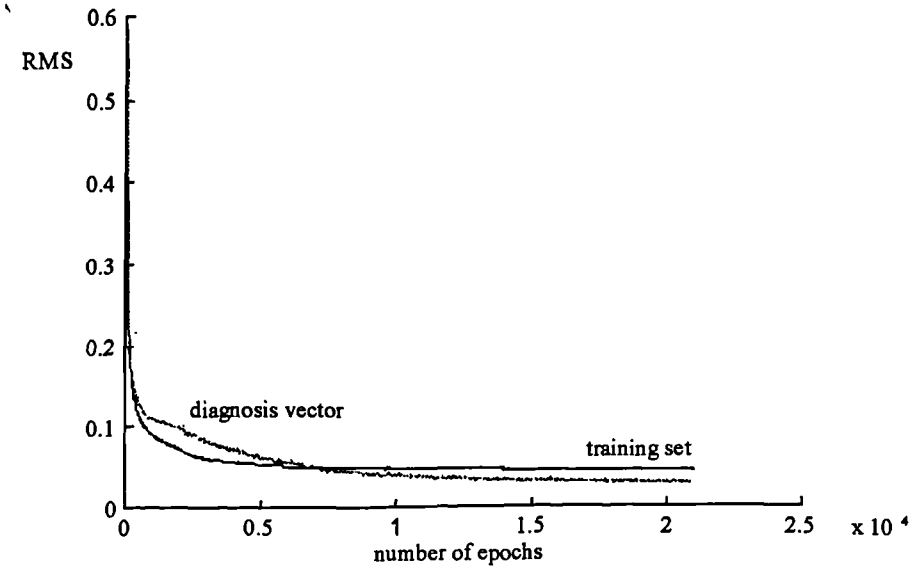


Fig. 3.16: a typical training of the regression net (cat. B)

3.5.2 Results and conclusions

400 diagnosis vectors have been used to test the actual capability of the system and 200 among them had a measurement too distorted.
Preliminary classification has always been performed successfully even in presence of a biased measurement.
All diagnosis vectors have been correctly classified by the first module (cat. A) and this confirms the suitability of NNs for classification tasks. The second module was not able to detect measurement biases all the times: 10 corrupted measurement vectors have not been recognised as such and the diagnosis has been carried out in those cases with the complete measurement set. Even so, the final calculation has been affected by a RMS of about 0,2. An important factor is the suitability of the available (in case remaining) measurements to detect the particular fault at hand. GPA accuracy can be increased by a choice of measurements suited to the fault being sought and the same applies to the neural approach presented here. In particular it can be shown that the measurement set available on the analysed engine is not optimal to detect all kinds of fault considered (Provost, 1995; Provost and Singh, 1995).

The average RMS for category A has been 0.07 and so the accuracy of the developed method can be considered acceptable from a diagnostic point of view. When a measurement is found to be affected by bias, the final RMS is usually larger than it would be by using the complete correct set of measurements. As expected, the RMS is usually larger in case of several non-zero performance parameters, apart from the amount of deterioration.
The results for category B have been even better, as no misclassification occurred and the average RMS has been 0.04. This shows that the system ability to approximate non-linear functions (moreover in a noisy and uncertain environment) is high and it can be exploited to produce estimations less affected by "smearing". Actually neural nets could be used to concentrate the subsequent analysis on a subset of performance parameters.

The proposed neural diagnostic method is able to quantify multiple faults affecting 3 out of the 4 basic engine components by using few and noisy measurements and can detect a measurement bias. The accuracy of the performance parameter estimation is high and the bias detection good. When a bias is not detected because of its low value, the system relies on its inherent robustness, being able to produce acceptable answers.
Novelties of the work with respect to previous studies are:

- quantification of the faults
- multi-fault capability
- SFDI in presence of engine faults as well
- application of a RAANN with novel second step training.

However, major drawbacks of the developed system are:

- the number of nets to be trained is large
- the size of the diagnostic system would further increase if more operating points were analysed
- training of the RAANNs is very long
- only single sensor faults can be coped with in general
- even though some kind of crosschecking is effected (by means of the RAANNs used after the classification), the system's structure is basically sequential. That means that an error in an early stage of the diagnosis can completely the prediction's accuracy
- no allowance has been made to account for noise and biases in the parameters used for setting the environment and the operating point of the engine.

Further comments on the suitability of NNs for gas turbine diagnostics will be given in section 3.7.

Since a correct and complete assessment of the health of the engine can be obtained only by tracking the performance parameter variations with time, the system's diagnostic capability should be tested with typical fault temporal patterns and further improvement could consist in the embodiment of dynamic Sensor Failure Detection, Isolation and Accommodation.

3.6 Neural network based sensor validation for gas turbines

The present section describes a work published by the author on the use of NNs for gas turbine sensor validation (Zedda and Singh, 1999b).
The diagnostic system presented in section 3.5 used AANNs for single SFDI in presence of engine faults. In particular, RAANNs with a modified training strategy were used. Main aim of the present work is to thoroughly test AANNs for SFDI (and in case even SFDIA) when:

- the parameters setting the environment and power condition are noisy
- the parameters setting the environment and power condition are noisy and biased
- more operating conditions are analysed simultaneously

Again, engine faults are present along with sensor faults. The following assumption has been done:

- one or at most two engine components can be faulty simultaneously. This entails that a maximum of four performance parameters can be affected by faults.

Diagnostics developed according to this assumption should be properly named ***fault diagnosis***, to distinguish it from the ***analysis of deterioration***, where a relatively large number of engine components are likely to be simultaneously faulty. In the work presented in section 3.5 both approaches were actually implemented.

The reason why this type of study has been carried out is that AANNs allow to somehow decouple the measurement validation and the performance estimation problems. Due to the features of self-supervised learning, sensor faults are detected and identified without explicit estimation of the engine component faults. However, knowledge of the engine fault location is necessary to be able to exploit the measurement redundancy. Usually primary aim of the diagnostics is just the isolation of the faulty engine components. So, unless a classification by means of NNs like the one proposed in 3.5 is adopted, at this stage of the diagnosis the faulty engine components have not been identified yet. Consequently, the only way to effect sensor validation by means of AANNs is to train ***a bank of NNs***, each one corresponding to a different location of the engine faults. If fault diagnosis instead of analysis of deterioration is required, the total number of combinations of fault locations is usually not huge. If training of the various AANNs were performed in reasonably short times, utilisation of the bank of AANNs in recall mode would allow both SFDI and engine component faults' isolation to be accomplished, without the performance parameters being calculated directly.
Thus, for the sensor validation system to be of some use in fault diagnosis, it is necessary to devise a quick way to train and use the nets in recall mode to perform SFDI. This has been accomplished in the present work.

3.6.1 Modification of backpropagation for training AANNs

Backpropagation is the most widely used training algorithm because of its simplicity and effectiveness. Nonetheless, it has its own drawbacks that are mainly the presence of local minima and the slow rate of convergence (see section 3.2). The occurrence of these general drawbacks, along with the need to tailor backpropagation to a wide range of

specific problems, have led to the development of a large number of domain-specific modification to the plain, simple learning algorithm. Here a version of the backpropagation is proposed, which is particularly suitable for training AANNs with data whose noise is statistically well known (in terms of standard deviations).
The standard backpropagation uses the following cost function:

$$E(n) = \frac{1}{2}\sum_{j=1}^{p} e_j^2(n) \tag{3.35}$$

where the output neuron's error is defined:

$$e_j(n) = d_j(n) - y_j(n) \tag{3.36}$$

As worked out in section 3.2.1, the updating rule for the weights is:

$$\Delta w_{ji}(n) = \eta \cdot \delta_j(n) \cdot y_i(n) \tag{3.37}$$

where the local gradient is defined as follows:

$$\delta_j(n) = e_j(n) \cdot \varphi_j'(v_j(n)) \tag{3.38}$$

The modified backpropagation, though, uses the following cost function:

$$E(n) = \sum_{j=1}^{M} \frac{\left(d_j(n) - y_j(n)\right)^2}{\sigma_j^2} \tag{3.39}$$

where σ_j is the noise standard deviation of the j-th measurement.
If a new definition for the error is given:

$$e_j(n) = \frac{d_j(n) - y_j(n)}{\sigma_j} \tag{3.40}$$

then the cost function is again formally defined as in (3.35). The standard backpropagation algorithm is still applicable, provided the following definition is used:

$$\delta_j(n) = \frac{e_j(n)\varphi'_j(v_j(n))}{\sigma_j} \tag{3.41}$$

The basic idea is to weight the output errors differently on the basis of the known noise standard deviations. The proposed approach turns out to be more effective than simply scaling the input-output patterns to normalise to a common standard deviation. Furthermore, a novel threshold logic is introduced as explained in the next section.

3.6.2 Thresholding and optimisation

When training is over, test set results can be collected to create statistics about the typical errors in the reconstruction of the noisy but fault-free input patterns.
In particular it is possible to calculate the standard deviations σ'_j of the errors of the predicted output with respect to the desired target. These values are found to be significantly smaller than the original measurement noise σ_j's. The ***noise reduction*** is different for different measurements, but the ratio σ'_j/σ_j is usually about 0.25.
Fixed a certain c (e.g. $c=3$), whenever the difference between the output and the input is found to be larger than $c\cdot\sigma'_j$, a faulty sensor is detected. Apart from the M thresholds set for every output (M is the number of measurements), an overall threshold is defined for the following function:

$$WRMS=\sqrt{\frac{\sum_{j=1}^{M}\frac{\left(z_{inj}-z_{outj}\right)^2}{\sigma'^{\,2}_j}}{M}} \tag{3.42}$$

where:

- z_{inj} is the j-th input value
- z_{outj} is the j-th output as produced by the net.

If a certain output value is found to be out of bound, the corresponding measurement is not necessarily the biased one (see section 3.4.2 and Mah, 1990). Given the number M_{bias} of biases assumed to be present, though, isolation of the faulty sensors can be carried out by minimising function (3.42) where the variable inputs are given by all possible combination $N_{\min}$ of M_{bias} out of M input values:

$$N_{\min}=\binom{M}{M_{bias}}=\frac{M!}{M_{bias}!\cdot(M-M_{bias})!} \tag{3.43}$$

For a given selection of M_{bias} inputs allowed to vary to minimise (3.42), the remaining inputs are held fixed. After all $N_{\min}$ minimisations are done, the one producing the lowest WRMS value should allow to both isolate and accommodate the biased input, provided all output errors are found to be below the corresponding thresholds as well. The minimisation technique used in this work is the Powell's method with discard of the largest descent directions (Press et al., 1992). This technique is suitable to minimise the error function due to its quadratic like shape in the proximity of the minimum. Fig. 3.17 shows the typical shape of the error function in the area where the minimum lies. Fig. 3.18 shows the corresponding isopotential contours.

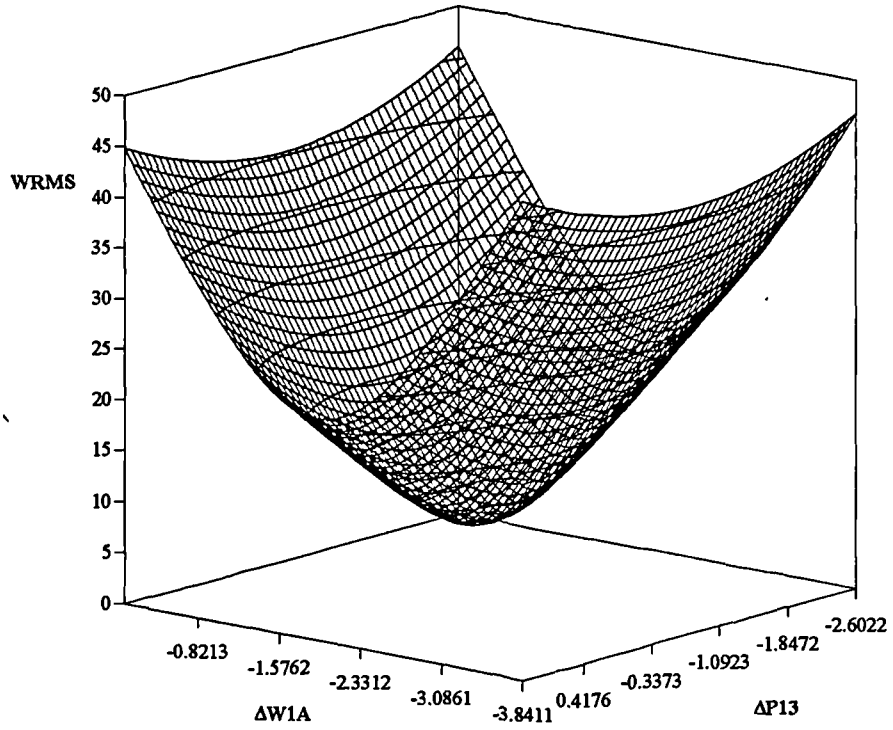


Fig. 3.17: shape of the WRMS function

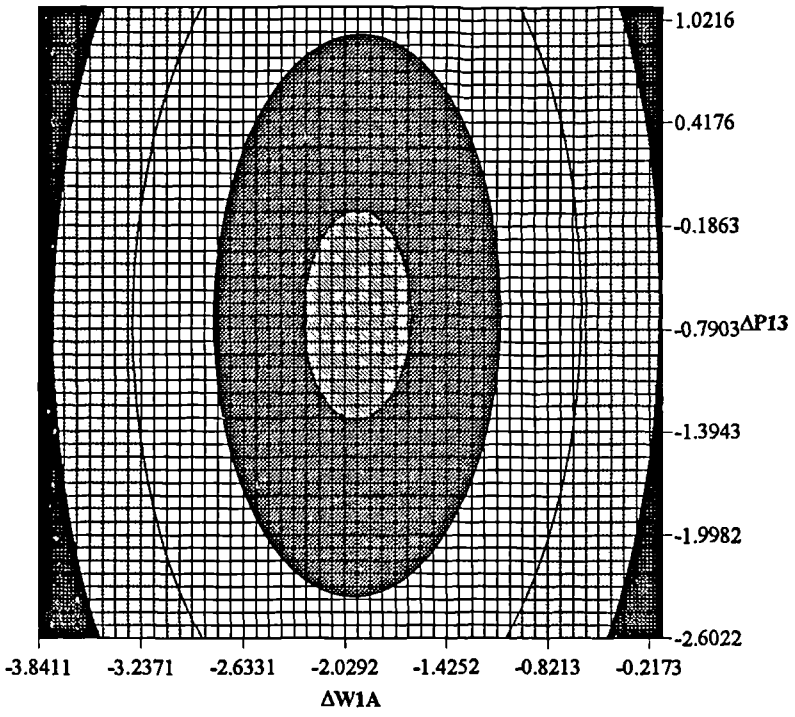


Fig. 3.18: isopotential contours of the WRMS function

Two points have to be made:

1. even though M_{bias} has to be set at the beginning of the minimisation, usually an estimation of the maximum possible number of biased measurements is available. Besides, an overestimation on M_{bias} will provide isolation of the faulty sensors and some largely noisy measurements as well.
2. minimisation of function (3.42) is rather straightforward. Few iterations of the Powell's methods usually allow to reach the minimum.

3.6.3 The engine and the approach

The engine with which the diagnostics is tested is a low by pass ratio turbofan, the EJ200, powerplant of the European Fighter Aicraft (EFA 2000). See section 4.3 for details. A Rolls-Royce accurate non-linear steady state performance simulation model (RRAP) is used to generate training and test data (see section 4.4 for details on the model). The engine's health is modelled by 6 components: inner and outer fan, HP compressor, HP turbine, LP turbine and propelling nozzle. 10 performance parameters are used to monitor the health of the engine components: outer and inner fan efficiency (η_{FANOUT}, η_{FANIN}), overall fan flow function (Γ_{FAN}), nozzle discharge coefficient (C_D), HPC, HPT, LPT efficiencies and flow functions (η_{HPC}, Γ_{HPC}, η_{HPT}, Γ_{HPT}, η_{LPT}, Γ_{LPT}). The instrumentation suite supposed to be available in test bed is made of 16 measurements: inlet total pressure and temperature (P_1 and T_1), spool speeds (N_H and N_L), outer fan exit total pressure and temperature (P_{13} and T_{13}), inner fan exit total pressure and temperature (P_{21} and T_{21}), inner fan mass flow (W_{21}), HP compressor exit total pressure and temperature (P_3 and T_3), LP turbine exit total pressure and temperature (P_5 and T_5), main burner fuel flow (W_{FE}), engine inlet airflow (W_{1A}), thrust (F). P_1 and T_1 are used to set the ambient condition, W_{FE} is given and F is measured. Thus 3 measurements are used to set the operating point and 13 to monitor the engine components. This is the typical instrumentation available for ***test bed analysis***. Real measurement noise levels are used (see section 4.10 for details).

In the present work, only one or two engine components are allowed to be simultaneously faulty (see section 3.6). However, as the location of the engine faults is unknown, a bank of NNs have to be used. For the engine considered, 21 nets have to be trained: 6 corresponding to single engine component faults and 15 corresponding to two engine faulty components. For every ***fault class*** a database is created by using the performance simulation code in synthesis mode: given the performance parameters the measurements are calculated and then noised. The maximum level of deterioration is 3%. The performance parameters uniformly span the allowed range of deterioration. Training and test sets are chosen randomly from the fault class database.

A key point is to understand the number of bottleneck neurons required and their meaning. To do so, it is necessary to reverse to the basic measurement equations that are available.

If the effect of both noise and biases is taken into account, the relationship between measurements and performance parameters can be written as follows:

$$\mathbf{z} = \mathbf{h}(\mathbf{x},\mathbf{w}) + \mathbf{b} + \mathbf{v} \tag{3.44}$$

where:

- $\mathbf{z} \in R^M$ is the measurement vector and M is the number of measurements
- $\mathbf{x} \in R^N$ is the performance parameter vector and N is the number of parameters
- $\mathbf{w} \in R^P$ is the vector of the environment and power setting measurements (e.g. inlet condition parameters and fuel flow) and P is the number of measurements
- $\mathbf{b} \in R^M$ is the bias vector
- $\mathbf{v} \in R^M$ is the measurement noise vector
- $\mathbf{h}(\cdot)$ is a vector-valued function representing the ideal measurements.

When performing analysis, $\mathbf{h}(\cdot)$ is usually provided by a simulation model; it is non-linear.

Eq. (3.44) states that when engine faults ($\mathbf{x}$) and operating condition ($\mathbf{w}$) are fixed, the measurements actually collected from the engine ($\mathbf{z}$) will differ from their ideal value due to noise ($\mathbf{v}$) and biases ($\mathbf{b}$). Eq. (3.44) is just like (2.3) with the environment and power setting parameters written explicitly.

$\mathbf{w}$ is affected by noise as well as biases like the other measurements:

$$\mathbf{u} = \mathbf{w} + \mathbf{b}_w + \mathbf{v}_w \tag{3.45}$$

where:

- $\mathbf{u}$ is the vector of measured values
- $\mathbf{w}$ is the vector of actual values
- $\mathbf{b}_w$ is the vector of biases
- $\mathbf{v}_w$ is the vector of noise.

Measurement equations (3.44) and (3.45) define the problem. The classic way to solve it is to use KF-based techniques to estimate $\mathbf{x}$, $\mathbf{w}$ and $\mathbf{b}$ given $\mathbf{z}$, $\mathbf{u}$ and statistics on $\mathbf{v}$ and $\mathbf{v}_w$.

It is worthwhile to notice that this definition of the problem is the same as the one that is used for the optimisation-based approach to the gas turbine diagnostics, as explained in chapter 4.

3 different kinds of test cases have been considered:

1. the environment and power setting parameters $\mathbf{u}$ are not affected by biases
2. the environment and power setting parameters $\mathbf{u}$ are affected by biases as well
3. more operating points are used for the sensor validation task.

The next 3 sections describe and analyse the approach and the results for the corresponding diagnostic problems.

3.6.3.1 No bias in the setting parameters

The parameters setting the operating condition of the engine are assumed to be noisy but not biased ($\mathbf{b}_w = \mathbf{0}$). Here the input vector $\mathbf{t}$ and the principal components $\mathbf{y}$ of eq. (3.28) are $\mathbf{z}$ and $\mathbf{x}$ respectively. Every net is made of 5 layers: 13 neurons in the input and in the output layer and a number of bottleneck neurons equal to the number (from 1 to 4) of performance parameters for the considered components. The number of neurons in the mapping and demapping layers is set to 20 when 1 or 2 bottleneck neurons are used, 30 when 4 neurons are used. A larger number of neurons in these two layers do not produce improvements in the performance of the net. The activation functions are the identity function for input and output layers, the hyperbolic tangent for the hidden layers. Input and output quantities are expressed by relative values with respect to the undeteriorated off-design condition as defined by the noisy environment and power setting parameters:

$$\Delta z_j = \frac{z_{\det j} - z_{undet\,j}}{z_{undet\,j}} \cdot 100 \tag{3.46}$$

where:

$$z_{\det j} = z_{\det j}(\mathbf{x}, \mathbf{w}) \tag{3.47}$$

$$z_{undet\,j} = z_{undet\,j}(\mathbf{u}) \tag{3.48}$$

Once the nets are trained, 2100 biased patterns are used to test the SFDI performance of the nets. For every bias pattern, the optimisations are carried out for all 21 fault classes. 2 and 4 biases are superimposed to the noisy input measurements. Their magnitude is set to 2%. In the 2 bias case 97%, in the 4 bias case 93% sensor faults are successfully detected and isolated. Better performance achieved when only 2 biases are present is due to the availability of larger redundancy.
Fig. 3.19 shows the typical learning curves for both standard and modified backpropagation. The modified backpropagation's training curve is smoother and the final error is smaller. In this case the net is designed to analyse data from a two faulty component class and the number of bottleneck neurons is 4. In the figure, the average Root Mean Square error vs number of epochs is plotted.

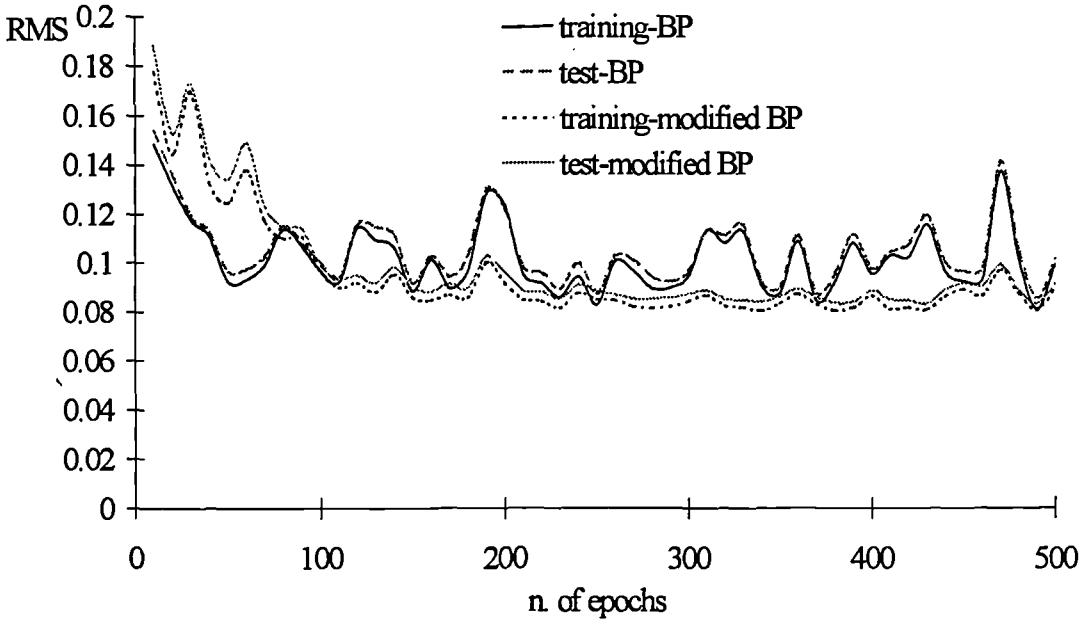


Fig. 3.19: comparison of training

The number of training patterns used is set depending on the number of required bottleneck neurons. The number ranges from 300 to 1000. 500 iterations are sufficient to reach an average RMS of about 0.09.

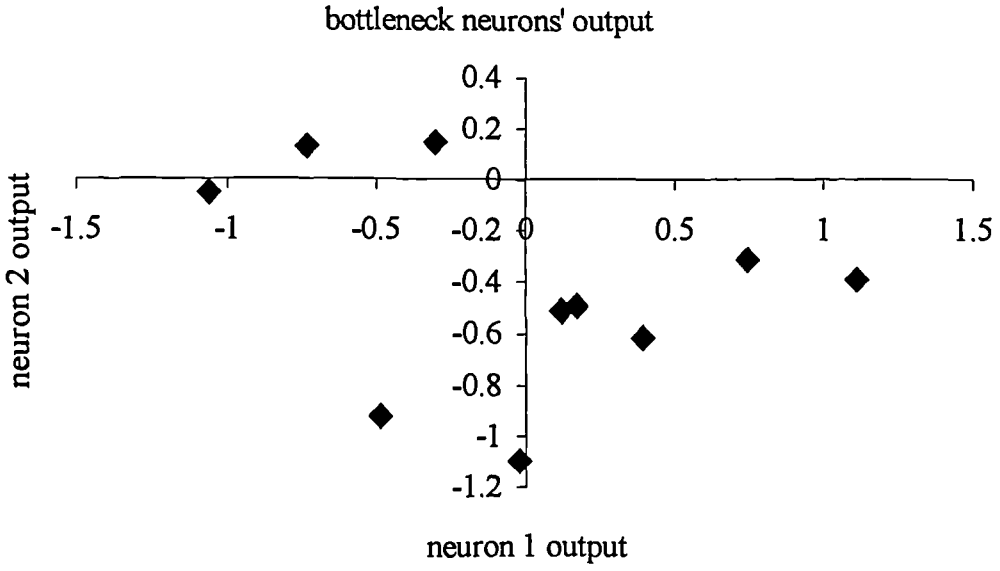


Fig. 3.20: principal components

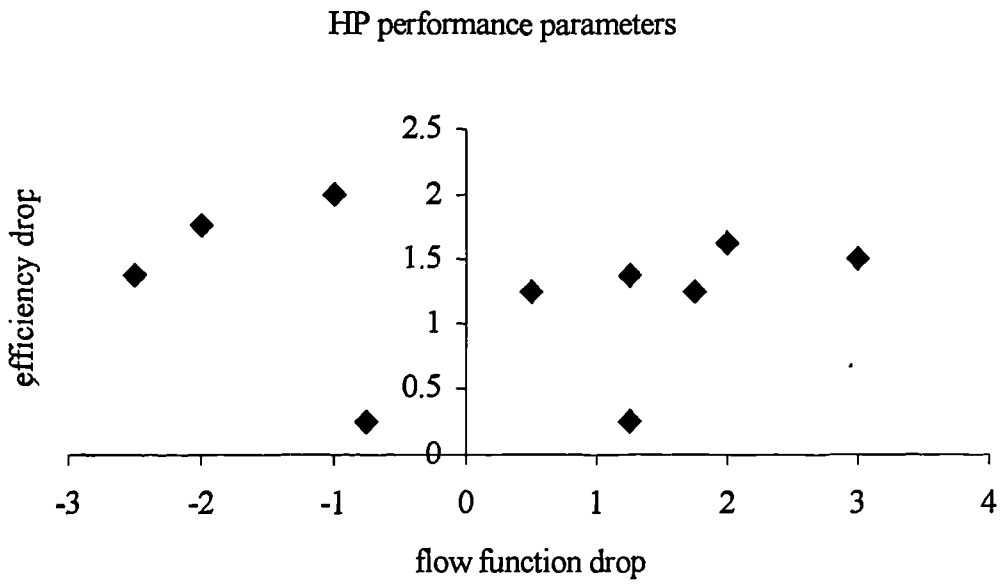


Fig. 3.21: performance parameter deltas

Fig. 3.20 shows the NLPCA carried out in the bottleneck layer for the case of a faulty HP compressor. The output values of the two bottleneck neurons are plotted for 10 patterns and comparison with the actual deterioration defined in fig. 3.21 by the efficiency and the flow function drops can be made. The quantities plotted in fig. 3.20 represent the principal component vector $\mathbf{y}$. Even though a certain distortion is present due to the measurement noise and the non-linearity of the mapping, fig. 3.20 and 3.21 show that AANNs perform NLPCA.
However, it is worthwhile to remark that:

- The mapping is non-linear and non-orthogonal: in general right angles are not mapped onto right angles
- the mapping from the input to the bottleneck layer is not unique. Two trainings are likely to produce different weights and then different mappings.

The minimisation-based technique should enable simultaneous isolation and accommodation. In the present work accurate accommodation is achieved only when the data used for training the nets are generated by assuming that no noise is affecting the environment and power setting parameters. When this noise is taken into account, the replaced values may not be accurate estimates of the biased measurements.

3.6.3.2 Biases in the setting parameters

The parameters setting the environment and power condition of the engine ($\mathbf{u}$) are affected not only by noise ($\mathbf{v}_w$) but also by biases ($\mathbf{b}_w \neq \mathbf{0}$).
In this case a reference value $\mathbf{u}$ is given for the actual $\mathbf{w}$ and the principal components will be related to $\mathbf{x}$ and $\mathbf{w}$. The input-output values are again defined by (3.46), but the

training and test sets are generated by varying the actual $\mathbf{w}$ inside a range defined by 50 times the noise standard deviations.

As 3 parameters are used to set the operating point of the engine, the number of bottleneck neurons will range from 4 to 7 depending on the number of fault affected performance parameters. 40-45 neurons are used in the mapping and demapping layers. Training is usually more difficult when the redundancy is smaller. 2100 patterns are used to test the SFDI performance of the proposed SFDI technique. .2 or 4 biases of 2% magnitude are superimposed to the noisy input vectors. In the 2 bias case 91%, in the 4 bias case 87% sensor faults are successfully detected and isolated. Again, the larger the redundancy the better the SFDI performance.

3.6.3.3 Multiple operating point SFDI

AANNs are used to carry out SFDI for a given faulty engine analysed at 2 different operating points. As explained in section 2.3, the amount of information extractable from the instrumentation can be maximised by collecting data for different operating points ($\mathbf{w}$) at the same fault condition ($\mathbf{x}$). However, as pointed out by Doel (Doel, 1994) the operating points should not be far from one another to minimise the effect of distortion on the component maps due to faults. For this reason, the two considered points are close in an area of high power. Apart from the different fuel flow, ambient pressure and temperature are different as well. In this case $\mathbf{z}_1$ and $\mathbf{z}_2$ will be the input vectors (input-output dimensionality=26). If P is the dimensionality of $\mathbf{w}$ and N_{perf} is the number of fault affected performance parameters, the number of bottleneck neurons is equal to $N_{perf}+2\cdot P$. It ranges from 7 to 10. 30-40 neurons are used in the mapping and demapping layers. Input-output vectors are scaled according to (3.46), where $z_{undet_j}(\mathbf{u})$ relates to the specific operating point.
Training is relatively easy in this case, due to the large redundancy available. 500 iterations are sufficient to get an RMS of about 0.09. The training sets are made of 2000-4000 examples due to the need for comprehensive mapping of the principal component space. 2100 are used to test the SFDI performance of the proposed technique. 2 or 4 biases (magnitude 2%) are superimposed to the input vectors. . In the 2 bias case 95%, in the 4 bias case 91% sensor faults are successfully detected and isolated.

3.6.4 Conclusions

The results of the SFDI testing are summarised in table 3.2 in percent successful sensor failure isolations. Appendix I contains some test samples for the various diagnostic cases.

The detectable trends are:

- an increase in the number of biases worsens the isolation performance because of the reduced redundancy that is available

- biases in the environment and power setting parameters make the isolation of monitoring measurement biases more difficult
- the use of more operating points improves the isolation's performance. It is worth noting, though, that the results above have been obtained with simulated data. If real data were used for testing, the actual performance would also depend on the map's distortion (see section 2.3).

	$M_{bias} = 2$	$M_{bias} = 4$
Case 1	97%	93%
Case 2	91%	87%
Case 3	95%	91%

Table 3.2: SFDI results

The main novelties introduced by the NN-based SFDI system are:
1. a new version of the backpropagation algorithm is proposed that is tailored for the autoassociation problem at hand
2. the actual advantage of the new backpropagation algorithm is that it can be coupled with a minimisation technique and a corresponding threshold logic
3. the second step training of the common AANN (or even of the RAANN) is substituted by an optimisation process.

The advantages of the SFDI system with respect to the one presented in section 3.5 are the following:
- the new training algorithm allows to reach slightly lower training errors faster
- the main benefit of the new training algorithm, though, is the high SFDI performance achieved by introducing a new objective function, the RMS weighted by the noise standard deviations as well as weighted thresholds. In the most difficult cases, the system was able to cope with 7 biases with an instrumentation set made of 16 sensors
- while the second step training of an AANN (or even a RAANN) is long and difficult, optimisation of the weighted RMS is straightforward from a numerical point of view
- all real effects due to measurement uncertainty are taken into account: biases are supposed to affect the environment and power setting parameters too
- the SFDI system has been developed for SFDI of a test bed, where the instrumentation can be supposed to be comprehensive. However, the same SFDI method could be used also with much poorer instrumentation sets, like the ones which are available for pass off tests or even on wing.

However, drawbacks of the system are:
- accommodation was not achievable with the required level of accuracy due to the complexity of the relationships involved and the large level of measurement noise
- the system has to be used to validate measurements before the performance parameters' estimation is carried out. As at this stage the location of faults is unknown, a bank of nets has to be used. This implies that a large number of nets have to be trained and then sequentially optimised. Therefore the system is little flexible and definitely more burdensome than classic estimation techniques used for SFDIA

(e.g. Kalman filtering) from a computational point of view. The situation is even worse when the method is applied to three-spool engines, because of the larger number of performance parameters and thereafter fault classes

- even though the SFDI accuracy obtained is good, it must be reminded that the magnitude of the biases detected and isolated was 2%. Even though this result represents a step ahead with respect to current NN-based SFDI techniques, the system certainly provides much worse results when tested with smaller biases (e.g. 1%). The notorious effect of undetected biases on the diagnostic accuracy suggests that an estimation technique completely reliant on this NN-based SFDI could suffer from inaccuracy in an environment characterised by a large number of biases.

CHAPTER 4

GAS TURBINE ENGINE AND SENSOR FAULT DIAGNOSIS USING OPTIMISATION TECHNIQUES

4.1 Introduction

Chapters 2 and 3 have shown there exist many different diagnostic problems and many techniques that can be used to deal with them.
As mentioned in section 1.3, the present work focuses on test bed analysis of development engines. A novel method has been developed and tested, that allows accounting for most of the measurement uncertainty effects that make application of GPA a difficult task. Although the technique has been purposely devised for development engines, further modifications have shown it can be applied to test beds fitted with just few sensors. Therefore, the method is applicable to pass off tests as well.
Two points have to be made:

- the relatively broad review concerning a wide range of diagnostic techniques, not necessarily aimed at test bed analysis, was required to glean concepts and methodologies which turned out to be extremely helpful when it came to invent a novel approach. The proposed method is actually the juxtaposition of many ideas and up-to-date numerical techniques.
- The reason why development engine test bed analysis has been chosen for a proper study work is that it requires a very detailed and accurate treatment of the problem. In the quest for new generation methods able to make performance analysis more effective and reliable, it was deemed important to set a standard of diagnostic accuracy, which can turn out to be useful in future development of simplified techniques. The availability of accurate performance analysis tools has long been due: when a development engine is in test bed, it is not uncommon that many days are spent trying to figure out the cause for a loss of performance. Usage of an analysis tool able to accurately sort out the problem would be very welcomed. Therefore, during the project the constraint as to the computing power utilised has been regarded as a secondary requirement. Nonetheless, although the proposed method is certainly more burdensome than the classic techniques from a computational point of view, it can be run in reasonably short times and so it can be considered a practical tool for accurate fault diagnosis

4.2 Test bed analysis of development engines

The detailed design and development programme of a gas turbine is a long process, which may typically take 3-7 years from inception to service entry. Development includes individual component tests and hundreds of hours of engine testing. On the basis of test results, several design modifications are introduced until the engine satisfies the original specifications. Optimisation of the results' interpretation and in conclusion thorough understanding of the engine's changing performance are highly desirable, as they reduce costs and speed up the development process.
From a diagnostic point of view, the distinctive features of development test bed analysis are:

- a comprehensive instrumentation set is available. A large number of sensors are usually fitted on the engine to monitor the components' performance
- an accurate, tailored performance simulation model is used for analysis
- only few engine components and sensors are likely to be simultaneously faulty. The spread of deterioration typical of service engines is not present here. The most common case is that one or at most two engine components are faulty in presence of a couple of biased measurements.
- particular emphasis has to be put on performance analysis accuracy. Isolation of faulty engine components and quantification of the drop in performance is of the utmost importance.

4.3 The EJ200 engine and instrumentation

The engine on which diagnostics has been developed is a two-spool military turbofan, the EJ200. It is the powerplant of the European Fighter Aircraft (EFA). It has been chosen for two main reasons:

- being a new engine, it could benefit from the diagnostics during future development phases
- it is sufficiently representative of the complexity of today's jet engines.

After development of the diagnostic method, tests have been done on a three-spool military engine as well, the RB199. See sections 4.11-4.12 for details.

The EJ200 is a low by-pass ratio mixed flow reheated turbofan. Two spools, it is 90 kN thrust class. Table 4.1 displays the cycle parameters.

by pass ratio	0.4
fan pressure ratio	4.2
overall pressure ratio	26
combat (max reheat) thrust	90 kN
max dry thrust	60 kN
thrust weight ratio	10

Table 4.1: EJ200 cycle parameters

The fan has 3 stages with fixed geometry, the HP compressor has 5 stages (1 variable stators' row). The two turbines are single stage. A convergent-divergent nozzle is employed.

The EJ200 engine health is modelled by 6 components and like all low by pass ratio engines a single total mass flow graph is used. The components are fan outer, fan inner, HP compressor, HP turbine, LP turbine and propelling nozzle. Fig. 4.1 shows the model's schematic. In the picture, the dashed line is used for components whose performance is defined by constant parameters that are not to be estimated.

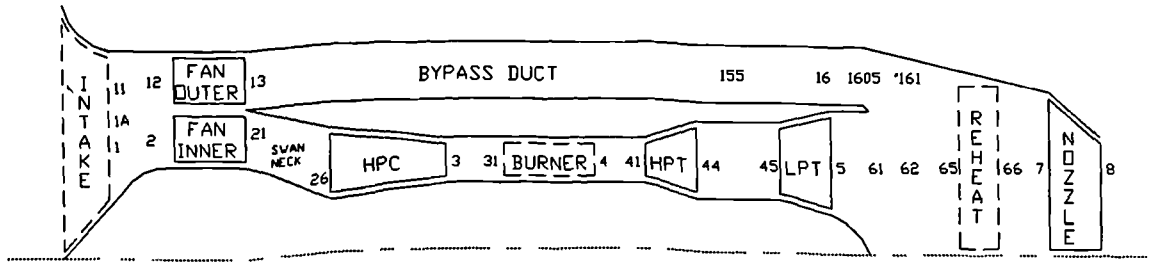


Fig. 4.1: schematic of the aero-thermodynamic model of the EJ200

Thus, the following performance parameters (10) express the health status of the engine:

- fan overall flow function (Γ_{FAN})
- fan outer efficiency (η_{FANOUT})
- fan inner efficiency (η_{FANIN})
- HP compressor flow function and efficiency (Γ_{HPC}, η_{HPC})
- HP turbine flow function and efficiency (Γ_{HPT}, η_{HPT})
- LP turbine flow function and efficiency (Γ_{LPT}, η_{LPT})
- propelling nozzle discharge coefficient (C_D).

13 measurements used for the analysis are those available in the test facility (*monitoring measurements*):

- engine inlet airflow (W_{1A})
- fan outer exit total pressure and temperature (P_{13}, T_{13})
- fan inner exit total pressure and temperature (P_{12}, T_{12})
- core inlet airflow (W_{12})
- HP compressor exit total pressure and temperature (P_3, T_3)
- LP turbine exit total pressure and temperature (P_5, T_5)
- net thrust (F_N)
- HP spool rotational speed (N_H)
- LP spool rotational speed (N_L)

3 parameters are used to set the operating point of the engine:

- main burner fuel flow (W_{FE})
- ambient total pressure and temperature (P_0, T_0).

4.4 The performance simulation model

The model used for simulation has been provided by Rolls-Royce. It is a non-linear steady state performance simulation program named RRAP. Full thermodynamic calculations, using enthalpy and entropy, are performed. In general gas properties are averaged one-dimensional values at any cross section of the gas path. Some two-dimensional values for pressure and temperature, enthalpy and entropy are used to approximate the effects of pressure and/or temperature distortion through the compression process. Combustion and expansion are strictly one-dimensional. Each rotating component is represented by a performance characteristic consisting of a tabulation of non-dimensional groups for flow, work, rotational speed and isentropic efficiency. Empirical data for nozzles and ducts, such as thrust, discharge coefficients and pressure losses are also included as tabulated data. The air system is represented by a series of bleeds removed from the gas stream between compressors or interstage. These bleeds are split and mixed where appropriate and returned to the main gas streams using flow and energy balances. The air system is modelled to the detail necessary to represent its effects on turbine operating gas temperatures, work done in the turbines and overall performance of the engine. Bleed flows are usually assumed to be a fixed percentage of the main gas stream flow. Extra detail is built into the models if the engine has a test data history in terms of bearing losses, heat transfer effects, customer bleeds and energy returns.
Internal calculations are performed in double precision and a parameter is provided, which allows setting the precision of the output values. For all subsequent calculations the parameter is set to $1 \cdot 10^{-6}$.

4.5 Coding

The codes developed during the project have been written in standard FORTRAN 77, according to the sponsor's requirement. Although an object-oriented approach would have been more suitable, the requirements for compatibility and portability prevailed. A modular structure has been used and the simulation code is called as a subroutine. Particular attention has been paid to make the code as portable as possible. Minimum use of built-in functions has been made. A large number of numerical subroutines have been obtained from Press et al. (1992). All codes developed for diagnostics are double precision and the numerical subroutines from Press at al. have been modified accordingly. All programs have been compiled with Digital Visual Fortran 5.0 (DVF 5.0).

4.6 Diagnostic requirements for test bed analysis

The diagnostic system to be developed had to fulfil a number of specifications that have been set at the outset of the project:

- one or at most two engine components are supposed to be simultaneously faulty. On this basis, a maximum number of fault-affected performance parameters is allowed ($N_{perf} = 1 \div 4$)
- a limited number of sensors are assumed to be simultaneously faulty: 2 or at most 4 monitoring measurements can be biased ($M_{bias} = 2$ or $M_{bias} = 4$). This assumption has been made in order to test the extent to which the method can be considered bias-tolerant. Actually, such a large number of faulty sensors are quite unlikely in a real test bed, where the occurrence of more than two simultaneous measurement biases is rather improbable
- all environment and power setting parameters are supposed to be simultaneously faulty. Again, this assumption has been made to stay on the safe side
- the performance parameters are allowed to depart from their off-design undeteriorated value at most by 3%. Flow functions are allowed to both increase and decrease.

These constraints seem to be sensible from a diagnostic point of view and from an analytical point of view they are definitely helpful in reducing the estimation inaccuracy and the "smearing".

It is worth noting that these specifications would apply to development test bed three-spool engines as well.
Eventually, it may be useful to remind the distinction between fault diagnosis and analysis of deterioration (see section 3.6). While the former assumes that just one or at most few engine components and/or sensors are simultaneously faulty, the latter assumes that a large number of them are faulty. Thus, the method developed performs proper fault diagnosis and not analysis of deterioration.

4.7 Gas turbine fault diagnosis as an optimisation problem

The fault diagnosis problem has been solved by minimising an objective function that has been built adequately. An introduction to the diagnostic problem, similar to the one presented in section 3.6.3, is given below.
If neither noise nor biases were present, the following relationship would hold:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) \tag{4.1}$$

where:

- $\mathbf{z} \in R^M$ is the measurement vector and M is the number of measurements
- $\mathbf{x} \in R^N$ is the performance parameter vector and N is the number of parameters
- $\mathbf{h}(\cdot)$ is a vector valued function.

$\mathbf{h}(\cdot)$ is provided by the simulation program and is non-linear. A further assumption for equation (4.1) to apply is the absence of model errors.
As measurement noise is present, eq. (4.1) must be modified as follows:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{v} \tag{4.2}$$

where $\mathbf{v}$ is the measurement noise vector.
In presence of biases, eq. (4.2) becomes:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{b} + \mathbf{v} \tag{4.3}$$

where $\mathbf{b}$ is the measurement bias vector.
Eq. (4.3) defines the relationship for a certain operating point. If the dependance on the operating point is written explicitly:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \mathbf{w}) + \mathbf{b} + \mathbf{v} \tag{4.4}$$

where $\mathbf{w} \in R^P$ is the vector of the environment and power setting parameters (e.g. inlet condition parameters and fuel flow) and P is the number of parameters.
Usually $\mathbf{v}$ is assumed to have a gaussian probability density function (pdf) and moreover to have independent components. Therefore, the joint pdf is the product of the indipendent pdfs:

$$p(\mathbf{v}) = \frac{1}{\left(\sqrt{2\pi}\right)^M} \prod_{j=1}^{M} \left(\frac{1}{\sigma_j}\right) e^{-\frac{1}{2}\sum_{j=1}^{M}\left(\frac{v_j}{\sigma_j}\right)^2} \tag{4.5}$$

where σ_j is the standard deviation of the j-th measurement.
It should be noted that $\mathbf{w}$ is affected by noise as well as biases like the other measurements:

$$\mathbf{u} = \mathbf{w} + \mathbf{b}_w + \mathbf{v}_w \tag{4.6}$$

where:

- $\mathbf{u}$ is the vector of measured values
- $\mathbf{w}$ is the vector of actual values
- $\mathbf{b}_w$ is the vector of biases
- $\mathbf{v}_w$ is the vector of noise.

Equations (4.4) and (4.6) define the problem.

The basic requirements for the objective function are as follows:

- it should be a measure of the consistency between actual and predicted measurements
- measurement noise should be accounted for
- measurement biases should be accounted for
- its minimisation should reduce the "smearing" effect
- evaluation of the function should not be too burdensome from a computational point of view.

It is worth noting that no statistical assumption is made about the performance parameters' pdf. The only assumption regards the measurement noise, which is usually statistically well known.

A classic choice for the objective function would be, given a certain operating point:

$$J(\mathbf{x}) = \sum_{j=1}^{M} \frac{\left[z_j - h_j(\mathbf{x})\right]^2}{(z_{odj}\sigma_j)^2} \tag{4.7}$$

where z_{odj} is the value of the j-th measurement in the ***off-design undeteriorated*** condition. The terms are conveniently expressed with respect to the reference condition. Minimisation of function (4.7) provides the maximum likelihood solution for the non-linear problem at hand (Bryson and Ho, 1975; Gelb, 1974).
Another suitable function is the absolute deviation:

$$J(\mathbf{x}) = \sum_{j=1}^{M} \frac{\left|z_j - h_j(\mathbf{x})\right|}{z_{odj}\sigma_j} \tag{4.8}$$

Eq. (4.7) and (4.8) are simple functions expressing the consistency between actual and predicted measurements. Measurement noise is taken into account by the standard deviations σ_j.
As uncertainty affects the environment and power setting parameters as well, function (4.7) could be modified as follows:

$$J(\mathbf{x},\mathbf{w}) = \sum_{j=1}^{M} \frac{\left[z_j - h_j(\mathbf{x},\mathbf{w})\right]^2}{\left(z_{odj}(\mathbf{w})\cdot\sigma_j\right)^2} \tag{4.9}$$

whereas function (4.8) would become:

$$J(\mathbf{x},\mathbf{w}) = \sum_{j=1}^{M} \frac{\left|z_j - h_j(\mathbf{x},\mathbf{w})\right|}{z_{odj}(\mathbf{w})\cdot\sigma_j} \tag{4.10}$$

Whenever the gaussian pdf (4.5) can be considered an accurate model of the noise, function (4.9) should be used. As a matter of fact, the actual measurement noise pdf may not be perfectly gaussian shaped, for a number of reasons. Then other objective functions, such as (4.10), are more suitable. An in-depth treatment of the subject is provided in section 4.10.1.

4.7.1 The Bayesian approach to optimisation

The choice of the objective function is obviously a key point for the success of diagnostics and should be guided by proper analysis in a statistical framework. Some investigations have been done to establish what the statistical theories suggest for the choice of the objective function. Thus, a brief introduction to the Bayesian approach to

the problem is given below, along with some conclusions on its applicability to the problem at hand.
For simplicity's sake, the following issues have not been taken into account:

- measurement biases
- uncertainty on the environment and power setting parameters.

In spite of these simplifications, useful hints can be attained with a simple theoretical analysis and few simulations.

As stated before (see section 2.2.1), the Kalman filter can recursively minimise an objective function when applied to a linear model. To fix the ideas, given a linear measurement vector equation:

$$\mathbf{z} = H \cdot \mathbf{x} + \mathbf{v} \tag{4.11}$$

where:

- $\mathbf{z}$ is the measurement vector
- $\mathbf{x}$ is the state vector, assumed to have a Gaussian probability density function
- H is a square matrix
- $\mathbf{v}$ is a zero-mean Gaussian noise vector

the Kalman filter can be shown (Bryson and Ho, 1975; Gelb, 1974) to be able to minimise the following objective function:

$$J(\hat{\mathbf{x}}|\mathbf{z}) = \int_{S1} \int_{S2} \ldots \int_{SN} L(\mathbf{x}, \hat{\mathbf{x}}) \cdot p(\mathbf{x}|\mathbf{z}) dx_1 dx_2 \ldots\ldots dx_N \tag{4.12}$$

where:

- N is the dimensionality of the problem. For simplicity both measurement space and state space are assumed to have the same dimension.
- $p(\mathbf{x}|\mathbf{z})$ is the conditional probability density function, otherwise named posterior probability density function
- S_j, $j = 1,2,\ldots,N$ are the domains of definition of the integrand functions
- $L(\hat{\mathbf{x}}, \mathbf{x})$ is the so-called *loss function*

when:

$$L(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{j=1}^{N} (x_j - \hat{x}_j)^2 \tag{4.13}$$

It can be shown (Gelb, 1974) that when the loss function is like (4.13) the vector minimising the objective function (4.12) is the conditional mean:

$$\hat{\mathbf{x}} = E(\mathbf{x}|\mathbf{z}) = \int_{S1} \int_{S2} \ldots\ldots \int_{SN} \mathbf{x} \cdot p(\mathbf{x}|\mathbf{z}) dx_1 dx_2 \ldots dx_N \tag{4.14}$$

If the technique was to be applied to gas turbine diagnostics, the following points should be made:

- the linear relationship (4.11) should be modified to account for non-linearities:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{v} \tag{4.15}$$

where $\mathbf{h}(\mathbf{x})$ is a non-linear function

- the assumption about Gaussian probability density functions for the performance parameters $\mathbf{x}$ seems to be arbitrary. A more sensible approach would be to assume a constant density function spanning a certain range. Fig. 4.2 shows the two probability density functions.

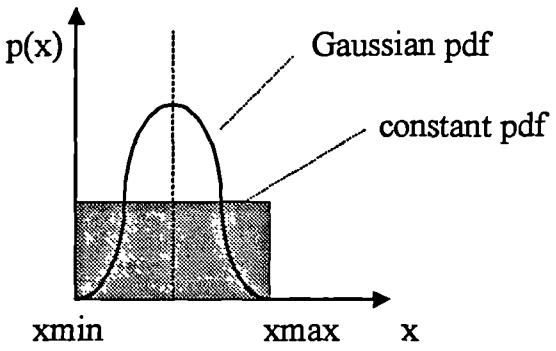


Fig. 4.2: comparison of probability density functions

Given the above assumptions, the outstanding question is how to calculate the conditional probability density function $p(\mathbf{x}|\mathbf{z})$. According to Bayes' theorem:

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{z})} \tag{4.16}$$

It can be shown (Bryson and Ho, 1975) that:

$$p(\mathbf{x},\mathbf{z}) = p(\mathbf{x},\mathbf{v}) \cdot \left| JJ^T \right|^{\frac{1}{2}} \tag{4.17}$$

where:

$$J = \frac{\partial[\mathbf{z} - \mathbf{h}(\mathbf{x})]}{\partial \mathbf{z}} = I \tag{4.18}$$

with I identity matrix.
Assuming x_j and z_j, $j = 1,2,..,N$ to be independent:

$$p(\mathbf{x},\mathbf{z}) = \prod_{j=1}^{N} p_j(x_j) \cdot \prod_{j=1}^{N} p_{vj}(z_j - h_j(\mathbf{x})) \tag{4.19}$$

As the performance parameters are assumed to have constant probability density function, the expression (4.19) can be rewritten as follows:

$$p(\mathbf{x},\mathbf{z}) = C \cdot \prod_{j=1}^{N} p_{vj}(z_j - h_j(\mathbf{x})) \tag{4.20}$$

where C is a constant.
Substituting (4.20) and (4.16) in (4.12):

$$J(\hat{\mathbf{x}}|\mathbf{z}) = K \cdot \int_{S1}\int_{S2}\ldots\int_{SN} L(\mathbf{x},\hat{\mathbf{x}}) \cdot \prod_{j=1}^{N} p_{vj}(z_j - h_j(\mathbf{x}))dx_1 dx_2 \ldots\ldots dx_N \tag{4.21}$$

where K is a constant.
Since the noise is zero-mean Gaussian:

$$p_{vj}(z_j - h_j(\mathbf{x})) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{[z_j - h_j(\mathbf{x})]^2}{2\sigma_j^2}} \tag{4.22}$$

where σ_j is the standard deviation of the j-th measurement.
(4.21) then becomes:

$$J(\hat{\mathbf{x}}|\mathbf{z}) = a \cdot \int_{S1}\int_{S2}\ldots\ldots\ldots\int_{SN} L(\mathbf{x},\hat{\mathbf{x}}) \cdot e^{\frac{\sum_{j=1}^{N}[z_j - h_j(\mathbf{x})]^2}{2\sigma_j^2}} dx_1 dx_2 \ldots\ldots dx_N \tag{4.23}$$

where a is a constant.
A Bayesian solution *of the problem would be the minimisation of function (4.23)* for any loss function $L(\mathbf{x},\hat{\mathbf{x}})$ chosen, given the assumptions of non-linear measurement equation and constant performance parameters' probability density functions.

A simple fictitious non-linear vector function has been chosen to test the effectiveness of the method. Rather than a proper minimisation of the function (4.23), a simple sampling of the function has been carried out in its domain. This allowed studying the effect on estimation accuracy of different loss functions. The outcomes of these preliminary simulations are:

- the average accuracy is good
- the choice of the loss function influences the results' accuracy and also the patterns of the error distribution (different levels of "smearing")
- the solution given by the quadratic function (4.13) is affected by a large level of "smearing"
- if the loss function is a sum of absolute values rather than squares, the "smearing" is reduced.

However, minimisation of the function (4.23) seems to be unfeasible whatever loss function $L(\mathbf{x},\hat{\mathbf{x}})$ may be chosen. It is reminded that:

- $\mathbf{h}(\mathbf{x})$ is usually a vector function whose evaluation is computationally burdensome
- multidimensional integration by numerical means is no easy task in general
- an important requirement for the objective function is that is has to be easily evaluated (see section 4.7).

In conclusion, for practical reasons the objective function should not be integral and therefore should be based on evaluations directly performed in the measurement space, although the accuracy may be affected.

4.7.2 Sensor validation

The functions described in section 4.7 can be modified to deal with measurement biases as well according to the following criterion: the presence of a bias will introduce inconsistency between actual and predicted measurements.

The way the optimisation-based diagnostic system handles measurement biases relies on the concept of ***relative redundancy***. If no bias affected the measurements, then minimisation of function (4.10) would estimate $(\mathbf{x},\mathbf{w})$ so that the equations used in the terms of the summation would be mutually consistent. The inconsistency due to a biased measurement would manifest itself with larger values of the objective function, since no $(\mathbf{x},\mathbf{w})$ can be found to correspond to predicted measurements fitting sufficiently well the real ones. The problem can be overcome by elimination in the summation of function (4.10) of the M_{bias} terms corresponding to the biased measurements. Then the remaining terms are mutually consistent and the optimised function will reach a low value. For the technique to apply, it is necessary that:

$$M - M_{bias} > N_{perf} + P \tag{4.24}$$

reminding that P is the number of environment and power setting parameters. It is also assumed that the $N_{perf} + P$ parameters are function of the $M - M_{bias}$ remaining measurements. The assumption is usually acceptable due to the coupling of the equations defined by (4.1).

This redundancy relative to the number of fault-affected performance parameters is required because if $M - M_{bias} = N_{perf} + P$ then any choice as to the identity of the biased measurements would produce a consistent solution. However if the equations defining the terms used in the objective function are more than the parameters to be estimated, the problem becomes overdetermined and the lowest value of the function should provide the more consistent solution.

In the case considered the relative redundancy is guaranteed by the assumption about the maximum number of faulty engine components, which is acceptable for fault diagnosis. In this respect it is worth noting that the sensor validation task can be feasible even if the number of measurements is smaller than the number of performance parameters ($M < N$), provided the inequality (4.24) still holds.

Due to the large level of measurement noise, the larger the L.H.S. with respect to the R.H.S. the better, as the redundancy is larger. The way just N_{perf} out of N performance parameters are allowed to vary is explained in section 4.9.3.

Typical SFDIA techniques proceed in three sequential steps: whenever a fault in the instrumentation set is detected, the faulty sensor is isolated and then the measurement is possibly accommodated. The approach described here is different and made of two steps: M_{bias} is chosen at the onset of the analysis, so that an M_{bias} number of measurements are not used in the objective function. The first step is the optimisation that produces an estimation of the performance parameters $\mathbf{x}$ and the environment and power setting parameters $\mathbf{w}$. As knowledge of possible measurement biases can be useful for future analyses, the second step consists of running the simulation code in synthesis mode to calculate, given the estimated $(\mathbf{x},\mathbf{w})$, the values of the M_{bias} measurements not used in the optimisation. If the difference between the actual and predicted measurements is larger than a threshold based on the noise standard deviations, then SFDIA is effected, otherwise the outcome of the analysis is that no bias is present and estimation of $(\mathbf{x},\mathbf{w})$ has been done by using the subset of measurements providing the best consistency.

Since the identity of the faulty sensors is unknown, a combinatorial search has to be done for every $(\mathbf{x},\mathbf{w})$ to find the selection providing the lowest value. If, for example, 4 biases are assumed to be present, every time the objective function $J(\mathbf{x},\mathbf{w})$ has to be evaluated during the optimisation procedure, the following set of functions are calculated by sequential elimination of 4 measurements:

$$J_{klmn}(\mathbf{x},\mathbf{w}) = \sum_{\substack{j=1 \\ j\neq k,l,m,n}}^{M} \frac{\left|z_j - h_j(\mathbf{x},\mathbf{w})\right|}{z_{odj}(\mathbf{w})\cdot\sigma_j} \tag{4.25}$$

The value assigned to the objective function will be:

$$J(\mathbf{x},\mathbf{w}) = \min_{k,l,m,n} J_{klmn}(\mathbf{x},\mathbf{w}) \tag{4.26}$$

In general, the number Q of evaluations of $J(\mathbf{x},\mathbf{w})$ to choose from is equal to the number of possible combinations:

$$Q = \binom{M}{M_{bias}} = \frac{M!}{M_{bias}!(M-M_{bias})!} \tag{4.27}$$

It is worth pointing out that the proposed objective function satisfies the requirements listed in section 4.7.

In order to master the principle of the sensor validation method just introduced, the following simple example can be helpful. A pipe is given, which is divided in two parts, A

and B (see fig. 4.3). A flow passes through it. Pressures at stations 1, 2 and 3 are measured. The health of parts A and B is quantified by X_A and X_B respectively. The three pressures are known functions of the health parameters:

$$P_1 = h_1(X_A, X_B) \tag{4.28}$$

$$P_2 = h_2(X_A, X_B) \tag{4.29}$$

$$P_3 = h_3(X_A, X_B) \tag{4.30}$$

An error in P_2 is assumed to be present.

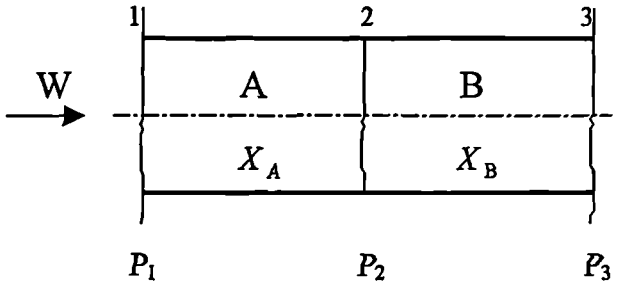


Fig. 4.3: sensor validation for a simple system

The classic approach to calculate the health parameters would consist of the minimisation of the squares:

$$J(X_A, X_B) = \sum_{j=1}^{3} \left[P_j - h_j(X_A, X_B)\right]^2 \tag{4.31}$$

A balance of the available equations-unknowns helps understand the problem. A look at equations (4.28) through (4.30) allows to say that 3 equations (one for each measurement) can be used to calculate the 2 health parameters. There is 1 redundant equation. If no bias is present, all 3 equations are mutually consistent (anyone of them can be expressed as combination of the other 2) and so the actual minimum of the chosen objective function (4.31) (i.e. zero) can be reached. If one of the measurements is biased, though, there is no mutual consistency and then the objective function (4.31) after optimisation will not reach the zero if all measurements are used. Detection of some sensor problem without identification of the faulty sensor is the only possible diagnostics. Moreover, the calculation of the health parameters is likely to be inaccurate.
A plain application of the sensor validation method just proposed suggests that the objective function should be:

$$J_i(X_A, X_B) = \sum_{\substack{j=1 \\ j \neq i}}^{3} \left[P_j - h_j(X_A, X_B)\right]^2 \tag{4.32}$$

$$J(X_A, X_B) = \min_i J_i(X_A, X_B) \tag{4.33}$$

However, the method does not work if just 3 measurements are available, because anyone of the 3 combinations obtained by elimination of 1 sensor (P_1 and P_2, or P_1 and P_3, or P_2 and P_3) can provide a solution with objective function equal to zero. This happens because when a measurement is not used in the calculation of the objective function there is no redundancy in the remaining set (2 unknowns, 2 equations). If, for example, another measurement is available, which can be related to any of X_A and X_B, say M_3, then elimination of 1 out of 4 measurements will leave a certain redundancy in the set used in the objective function: 2 unknowns are calculated by using 3 measurements. If the biased measurement is used in the objective function, then inconsistency will produce a larger value of the function, whereas when the biased measurement is excluded the minimum value will be zero. The important thing is that the number of measurements used to build the objective function be larger than the number of parameters to be calculated (at least one more), so that the redundancy enables checking on the self-consistency of the measurement set.
When dealing with engines, the 2 following points have to be made:

- noise is taken into account by weighting the terms to be added in the objective function. The minimum value will not be zero but when the minimisation is finished a simple check on the magnitude of the objective function can suggest if there is anything wrong with the analysis
- a large redundancy is usually available as the number of fault-affected performance parameters is assumed to be significantly smaller than the number of sensors. In the EJ200 at most 4 performance parameters and 3 environment and power setting parameters have to be estimated by using 13 measurements.

Environment and power setting parameter biases are dealt with in a different way, as they basically affect most of the terms in the summation. A bias in any of these parameters is likely to increase the value of most terms and therefore of the overall sum. Whereas a two step SFDIA is carried out for biased measurements, SFDIA is automatically performed for the environment and power setting parameters, as they have to be guessed by the optimiser.

Fig. 4.4 shows the layout of the diagnostic system. In the figure the input to the optimiser is highlighted: apart from the measurements, statistics of measurement noise and a maximum number of biases have to be entered. No assumption is made on the performance parameters' probability density function.

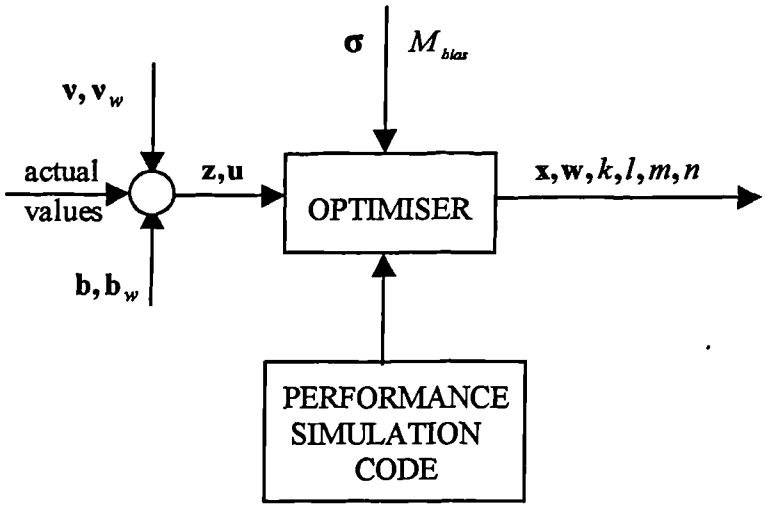


Fig. 4.4: schematic of the diagnostic system

4.7.3 Requirements for the optimisation technique

A large level of inaccuracy is likely to affect the solution if all performance parameters are allowed to be responsible for shifts in the measurements that are actually due to engine faults located in one or two components. Application of the constraint on the maximum number of faulty engine components enables reduction of inaccuracy and "smearing". Moreover it makes viable to use the sensor validation technique based on relative redundancy. Therefore a key point is the development of an optimisation technique able to:

- minimise a non-smooth function such as the one defined by (4.25) and (4.26)
- apply the constraint on the maximum number of fault-affected performance parameters (N_{perf}).

Having defined the problem, a suitable optimisation technique must be found or even devised if necessary. In the sequel a brief foray into optimisation is made with focus on the minimisation task which is needed for the diagnostic problem at hand.

4.8 Optimisation techniques

Optimisation methods can be classified according to various criteria. In the present work two different types of optimisation techniques have been used: calculus-based (or conventional) and evolutionary techniques. Evolutionary optimisation techniques have been introduced recently and are based on principles more or less directly derived from nature and its adaptation mechanisms. Calculus-based methods are more widespread and boast a very large number of applications.

A more rigorous and comprehensive classification of optimisation techniques is given below:

1) calculus-based
2) enumerative
3) random.

A brief qualitative distinction among the different search methods is given below.

1) *Calculus-based* optimisation techniques are widely used and can be:
   a) *indirect*: the objective function's gradient is set to zero and the resulting set of non-linear differential equations is solved
   b) *direct*: the solution is searched by moving towards directions somewhat related to the local gradient. The methods described in section 4.8.1 fall in this class.

   These optimisation techniques suffer from the following drawbacks:
   1) they are local in scope: the solution reached at the end of the search process is largely dependent on the start value. The absolute minimum value of the objective function is likely to be skipped unless the initial condition is not accurate enough. If the function is multi-modal, local minima are likely to be arrived at.
   2) they are applicable only when the functions involved are well behaved[1]
   3) most of them require calculation or at least estimation of the derivatives. Even though the functions involved in the minimisation process can be well behaved, the calculation of the derivatives may be difficult and thereby possibly to avoid.

   In conclusion, calculus-based optimisation techniques are definitely not robust.
2) *Enumerative* optimisation techniques work by discretising the space of search and by evaluating the objective function in all the selected points. The obvious drawback of this search method is its lack of efficiency. Even the most sophisticated enumerative technique (Dynamic Programming) breaks down on problems of moderate size, being affected by the so-called "curse of dimensionality".
3) *Random* optimisation techniques can be further classified as follows:
   a) *purely random* techniques: in the long run, they cannot be expected to perform better than enumerative schemes
   b) *randomised* techniques: in this case a search procedure uses random choice as a tool to guide a highly exploitative search through a coding of the parameter space. Basically random choice is used as a tool in a directed search. Genetic Algorithms (GAs) are an example of this type of optimisation technique.

Fig. 4.5 shows an example for which conventional optimisation techniques are likely to fail.

In the quest for suitable optimisation techniques, a key point has to be reminded: the objective function is "complex". This means that $J(\mathbf{x},\mathbf{w})$ is not the classic quadratic-shaped function and therefore its optimisation is likely to be difficult. In particular the presence of absolute values and the mechanism used to perform sensor validation make it non-smooth. Moreover, the effect of varying the environment and power setting parameters is likely to be dominant, as they affect most of the terms in the summation both in the denominator and numerator.

[1] From this point onwards, the term well behaved will be used for functions that are continuous with continuous derivatives up to the degree required

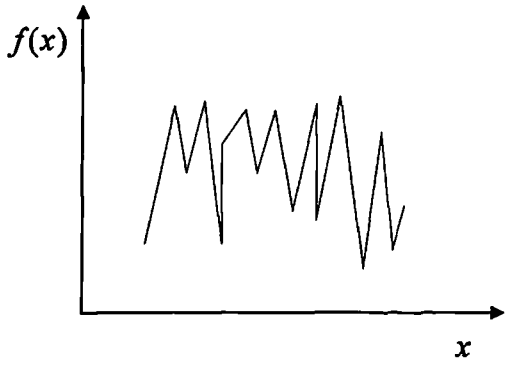


Fig. 4.5: a highly non-smooth function

In the light of the features of the multi-dimensional surface (or hypersurface) of $J(\mathbf{x},\mathbf{w})$ in the space of the performance parameters and the environment and power setting measurements, the following observations can be done:

- the technique should have some capability to deal with local minima, which the hypersurface could be spread with. If a strictly ***local search*** technique is employed, convergence to the global minimum is strongly dependent on the initial guess. Although it is common practice to repeat the optimisation a number of times with different initial values, a ***global search*** technique is to be preferred in that it should be able to cover a much wider area in the search space and then refrain from getting trapped in a local minimum. Utilisation of a global search technique, though, must not lead to a loss of accuracy, which is the primary diagnostic requirement
- some optimisation techniques rely on the calculation of gradients to reach the minimum. They can be used when the objective function is well behaved and are usually fast, accurate and local in nature. The non-smoothness of the objective function at hand does not allow utilisation of gradients for the optimisation. Thus suitable gradient-free methods have to be searched for.

In the sequel the different optimisation methods considered are analysed.

4.8.1 Conventional optimisation techniques

A first attempt to solve the optimisation problem has been through the use of an easy and relatively robust technique, named the ***Downhill Simplex method***, developed by Nelder and Mead (Press et al., 1992). At first the constraint on the maximum number of faulty engine components has not been applied to keep things simple. The method is not gradient-based and not very efficient in terms of the number of function evaluations required. A brief explanation of the way it works is given below.

A simplex is the geometrical figure consisting, in N dimensions, of $N+1$ points (or vertices) and all their interconnecting line segments and polygonal faces. In two dimensions, a simplex is a triangle. In three dimension, a simplex is a tetrahedron. In general, only non-degenerate simplexes, i.e. those which enclose a finite inner N-

dimensional volume, are of interest for the optimisation. If any point of a non-degenerate simplex is taken as the origin, then the N other points define vector directions that span the N-dimensional space. Whereas in one-dimensional optimisation it is possible to bracket a minimum, there is no analogous procedure in multidimensional space. The only thing possible is to give the algorithm a starting guess, that is an N-dimensional vector of the independent variables. The algorithm is then supposed to make its own way downhill through the N-dimensional topography until it reaches a minimum. The Simplex method must be started not just with a single point, but with $N+1$ points defining an initial simplex. If one of these points is regarded as the initial one $\mathbf{P}_0$, the other N points are chosen as follows:

$$\mathbf{P}_i = \mathbf{P}_0 + \lambda \cdot \mathbf{e}_i \tag{4.34}$$

where:

- $\mathbf{e}_i$ is the i-th unit or canonic vector
- λ is a constant which is a guess of the problem's characteristic length scale.

After initialisation, the method takes a series of steps, most of which just move the point of the simplex with the largest value of the function through the opposite face of the simplex to a lower point. The steps are called reflections and are constructed to conserve the volume of the simplex and hence maintain its non-degeneracy. When it can do so, the method expands the simplex in one or another direction to take larger steps. When a "valley floor" is reached, the simplex constracts itself in the transverse direction and tries to ooze down the valley. If there is a situation where the simplex is trying to "pass through the eye of a needle", it contracts itself in all directions, pulling itself around its lowest point (i.e. the best one). The algorithm is stopped when the decrease in the function value is fractionally smaller than a prescribed tolerance. It is common practice to restart the multidimensional minimisation at the point where it claims to have found a minimum.

The method has been used to optimise (4.25)-(4.26). The constraint on the maximum number of fault-affected performance parameters has been neglected at first.
The remarks to be made are:

- different λ's are used for different dimensions
- the optimisation algorithm is able to reach a minimum fairly quickly
- the minimum is usually quite far from the desired one. This can be checked on the final estimation error and the selection of faulty sensors.
- the algorithm finds it difficult to converge to a point characterised by faulty sensors different from the ones the optimisation process started with. This may happen because the algorithm is rather local in nature
- restarting the algorithm does not usually lead to other minima
- it is difficult to apply constraints on the range of variation of the parameters. As a matter of fact, constraining both the performance and the environment and power setting paramaters to vary inside a given range can be very useful. In particular, well-defined ranges can be set for the performance parameters (3% according to design requirements). If the algorithm is let free to vary the parameters, some of them easily slide out of the correct range and the diagnostic accuracy of the corresponding

solution is largely affected. Two methods have been applied to circumvent the problem:

1. whenever a parameter exceeds its limit, it is assigned the limit value itself. This method is not effective in that it modifies the internal rules of shaping of the simplex. The results obtained with the method are not accurate enough
2. whenever a parameter exceeds its limit, the objective function is augmented by a fixed amount. This method, named of the penalty function, provides better results than the previous one. The objective function becomes:

$$G(\mathbf{x},\mathbf{w}) = J(\mathbf{x},\mathbf{w}) + L \cdot N_{viol} \tag{4.35}$$

where:

- N_{viol} is the number of parameters exceeding the limit
- L is a constant.

However, the choice of the L influences the working of the algorithm very much and is a matter of tuning. Too large a value of L may prevent the algorithm from exploring areas which can be close to the minimum searched for, too small a value may lead to ultimately violate the constraints.

In conclusion, the Simplex method could be used along with a global search technique to locally refine the solution, provided an effective way is devised to deal with the constraints on the range.

Another conventional optimisation technique has been tested: the ***Powell's direction set method*** (Press et al., 1992). The method can be enclosed in the class of techniques that are supported by a one-dimensional minimisation method (usually the Brent technique). Given a start point $\mathbf{P}$ in the N-dimensional space, a trajectory, locally defined by a direction vector $\mathbf{n}$, is built according to some criterion. Then the objective function can be minimised along the trajectory by a one-dimensional method. It must be pointed out that one-dimensional optimisation is rather straightforward: bracketing the minimum and reaching it is relatively easy.
Different methods relate to different ways of choosing the trajectory.
A straightforward method is to choose the N axis directions as the trajectories on which to apply the one-dimensional optimisation: the function is optimised along the first axis and from the reached minimum a new optimisation is carried out along the second axis and so on. Cycling through the whole set of directions is stopped when the function stops decreasing. The method is easily implementable and works well for many functions. However, if the objective function is characterised by a narrow valley not aligned with any of the axes, the optimisation will go through a very large number of tiny steps. More generally, in N dimensions, if the function's second derivatives are much larger in magnitude in some directions than in others, then many cycles through all basis vectors will be required in order to get anywhere.
What is needed is a better set of directions than the basis vectors. That is striven for by the so-called *direction set methods*, where prescriptions for updating the set of directions as the method proceeds are given, attempting to come up with a set which:

1) either includes some number of "non-interfering" directions with the special property that optimisation along one direction is not spoilt by subsequent optimisations along another, so that interminable cycling through the set of directions can be avoided
2) or includes some directions aligned with a narrow valley .

Below some details of the two techniques are given.

1) The first strategy is realised by Powell's quadratically convergent method (Press et al., 1992). N almost "non-interefering" directions can be found and the algorithm converges quadratically to the minimum, i.e. N line optimisations are sufficient to reach the minimum of a quadratic objective function.
The algorithm proceeds as follows:

- the set of direction vectors $\mathbf{u}_i$ are initialised to the basis vectors:

$$\mathbf{u}_i = \mathbf{e}_i \qquad i = 1,\ldots,N \tag{4.36}$$

The following sequence of steps is repeated until the objective function stops decreasing:

- the *starting position is saved as* $\mathbf{P}_0$
- for $i = 1,\ldots,N$, $\mathbf{P}_{i-1}$ is moved to the minimum along direction $\mathbf{u}_i$ and the new point is called $\mathbf{P}_i$
- for $i = 1,\ldots,N$, $\mathbf{u}_i \leftarrow \mathbf{u}_{i+1}$
- $\mathbf{u}_N = \mathbf{P}_N - \mathbf{P}_0$
- $\mathbf{P}_N$ is moved to the minimum along direction $\mathbf{u}_N$ and the new point is called $\mathbf{P}_0$

If the function is not exactly quadratic, after N line optimisations the point reached will not be the minimum, but close to it. However, repeated cycles of N line optimisations will produce convergence to the actual minimum. As a matter of fact, though, Powell's quadratically convergent method tends to produce sets of directions that fold up on each other and become linearly dependent. Once this happens, the algorithm finds the minimum of the function only over a subspace of the full N-dimensional space and the corresponding solution is inaccurate. A number of ways have been proposed to cope with this problem, but none of them seems to improve convergence substantially, especially when the objective function is far from being quadratic.

2) The second strategy is more suitable for optimisation of functions characterised by long, narrow and twisty valleys. The basic idea of the method, devised by Powell, is still to take $\mathbf{P}_N - \mathbf{P}_0$ as a new direction, as average direction moved after trying all N possible directions. However, the old direction along which the objective function made its largest decrease is discarded. This choice is made because the best old direction is likely to be a major component of the newly added direction. Dropping the best old direction gives the best chance of avoiding a buildup of linear dependence. This method, properly refined with minor modifications, is ***Powell's direction set method with discard of the largest decrease direction***.

The Powell's algorithm with discard of the largest descent direction has been used to optimise function (4.25)-(4.26). Again, the constraint on the maximum number of fault-affected performance parameters has been neglected at first.
The results suggest that:

- the algorithm finds it difficult to handle the non-smooth surface of the objective function. If absolute values are substituted by squares, the non-smoothness is less pronounced, especially in the area surrounding the minimum. In fact, the optimisation turned out to be easier
- even with squares, for some test cases the algorithm is not able to provide accurate results. This may occur because of the non-smoothness of the objective function due to the sensor validation mechanism. A large number of solutions seem to have converged to local minima when the selection of faulty sensors is wrong
- if no constraint is applied to the range of variation of the parameters, the algorithm is likely to wander off the allowed area and induce problems of convergence of the performance simulation model (e.g. the model is required to evaluate the engine's performance for a 20% drop in efficiency)

Application of the two methods for constraining parameters already used for the Simplex technique has produced similar kind of results.
In conclusion, the Powell's technique does not seem to be suitable for the required optimisation. Furthermore, due to its lack of robustness and inability to deal with the objective function non-smoothness, it would be ineffective if used as a local enhancement to a global search method, unlike the Simplex technique.

4.8.2 Evolutionary optimisation techniques

The previous section has shown the intrinsic difficulties encountered by conventional techniques when applied to the optimisation-based diagnostic problem at hand. Evolutionary methods have then been explored, implemented and tested to ascertain their suitability to optimise the required objective function. These optimisation procedures, introduced recently and applied to an ever-increasing number of fields, show interesting properties. However, their application is not straightforward because of the following reasons:

- there exist a large number of different techniques and the choice of one of them can be crucial
- a theoretical analysis, which could be helpful to select some methods and discard others before implementation, is often not available
- the field of evolutionary optimisation is relatively young and a large number of key issues have not yet been addressed properly.

Even the nomenclature can be somewhat obscure. For instance, Genetic Algorithms (GAs) are a typical evolutionary optimisation technique. More than that, they are the most widespread evolutionary technique. However, the approach pursued in the present dissertation goes beyond the classic GA. Therefore, the term ***Evolution Programs*** (***EPs***) is here used to define any evolution-based system.

Although EPs include GAs as well, a better understanding of the matter is achievable by first introducing plain GAs and then EPs. After this brief foray into evolutionary

optimisation techniques, the real-coded GA developed for fault diagnosis of engine end sensor faults is explained along with results and the corresponding analysis.
Eventually, Evolution Strategies (ESs) are introduced. They are particular forms of EPs that show properties regarded as interesting for the problem at hand. Their application to the diagnostic problem at hand is shown as well.

4.8.2.1 Genetic Algorithms

GAs are currently used for a number of tasks, ranging from parameter optimisation to machine learning. However, here the focus is on optimisation. Nonetheless, there exist many different types of GAs able to perform optimisation.
A broad distinction can be made depending on the presence of coding:

- binary GAs do not work with the optimisation parameters directly, but with a binary representation of them
- real-coded GAs directly work in the real parameter space.

The first ones can be regarded as the classic ones, for whom a theoretical treatment has been developed. Therefore, even though in the present project real-coded GAs are used, binary GAs are introduced and analysed first.

4.8.2.1.1 Binary GAs

Binary GAs are search procedures that differ from the conventional optimisation methods in four ways:

- they work with a coding of the parameters, not the parameters themselves: the parameters are coded as a finite length string. Usually a binary representation is chosen
- they search for the optimal solution from a population of points, not a single point: this makes it easier to escape from local minima areas
- they use payoff (objective function) information: no use of derivatives is required. This enlarges the area of applicability of GAs very much
- they use probabilistic instead of deterministic transition rules.

If the real parameter used to optimise the objective function is within the range $[LB,UB]$, where LB is the lower bound and UB is the upper bound, it can be linearly mapped onto the range $[0,2^l]$, which is covered by a binary number with l digits. Mapping from a segment of the real axis onto a range of integer numbers requires a discretisation of the original segment. Increasing the number of digits used in the binary representation allows increasing the representation precision, but automatically increases the dimensionality of the problem.

The most simple GA is made of three operators, which are described below to give an idea of the way GAs work. A population of points is selected in a random way and input parameters are coded to finite length strings.

a) ***Reproduction or selection*** is a process in which individual strings are copied according to the values of their *fitness function*. The fitness function is related to the objective function. For instance, if the optimisation is a maximisation and the objective function in non-negative, the fitness function is simply coincident with the objective function. If the optimisation is a minimisation, the problem is changed to maximising a fitness function that is a simple decreasing function of the objective function (e.g. a linear function with negative derivative). As will be pointed out later, a requirement for the fitness function is that it be positive. Strings with high fitness value are more likely to breed, whereas low fitness strings are likely to die. A simple measure of the probability of survival can be given by the following formula:

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{4.37}$$

where:

- P_i is the probability of the i-th string to be replicated
- f_i is the fitness value of the i-th string
- N is the number of strings.

P_i represents the expected number of offspring of the i-th string. Various selection algorithms can be used to produce an integer number of offspring, hopefully as close as possible to the expected value as calculated by (4.37). See section 4.9.1 for details.

b) ***Crossover*** is a process to randomly create new strings from the current population. All strings are paired off randomly and for each pair a point of crossover is randomly chosen. Then the elements of the two strings located after the chosen position are swapped, as shown in fig. 4.6.

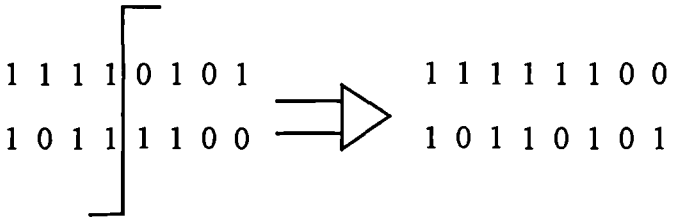


Fig. 4.6: simple crossover

c) ***Mutation*** is a way of randomly changing the value of a string position in order to avoid losing some potentially useful kind of string. This operator is of secondary importance and the bulk of the optimisation power is due to the first two operators.

Even though the described GA is very simple and easy to implement, it is able to tackle quite complex optimisation problems. It is necessary noting that:

- more complex operators can be used, especially when the nature of the optimisation problem at hand is analysed

- a quite detailed mathematical analysis of the optimisation performance of binary GAs can be done (Goldberg, 1989).

The effectiveness of binary GAs as an optimisation tool has been demonstrated by a very large number of applications to a wide range of problems. However, a significant contribute to the diffusion of the method has also been given by the development of a theory able to explain the power of search of binary GAs. In the following the main concepts are introduced and briefly discussed.

A *schema* is a similarity template describing a subset of strings with similarities at certain string positions. If a binary coding of the input parameters is assumed, three symbols can be used to represent a schema: 1, 0 and *. The last symbol represents a "don't care" option, which means that in that position either 1 or 0 may be present. For instance, the string 100010 can be represented by the schema 10**10 and 10***0 and 10**1* and so on.
The *order* of a schema H, denoted by $o(H)$, is simply the number of fixed positions (in a binary alphabet, the number of 1's and 0's) present in the template. For example, the order of the schema 011*1** is 4 (i.e. o(011*1**)=4).
The *defining length* of a schema H, denoted by $d(H)$, is the distance between the first and the last specific string position. For example, the schema 011*1** has defining length 4 (i.e. d(011*1**)=4).
Given these definitions, the ***Fundamental Theorem of Genetic Algorithms*** can be proved. If the following definitions are given:

- $m(H,t)$ is the number of examples of schema H at time t
- $f(H)$ is the average fitness of the strings representing schema H
- $\bar{f}$ is the average fitness of the entire population
- p_c is the probability of crossover
- p_m is the probability of mutation
- l is the string length

the theorem states that a particular schema H receives an expected number of copies in the next generation under reproduction, crossover and mutation according to the following equation:

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{d(H)}{l-1} - o(H) p_m \right] \tag{4.38}$$

The theorem means that high-performance, short-defining length, low order schemata receive at least exponentially increasing numbers of trials in successive generations. This occurs because reproduction allocates more copies to the best schemata and because simple crossover does not disturb short defining length schemata with high frequency. Since mutation is assumed to be quite infrequent, it has little effect on these important schemata. The exponential form of this allocation turns out to be a rational way to allocate trials, as it connects to the optimal solution of the two armed bandit problem, which is a classic decision problem (Goldberg, 1989). High-performance, short-defining-length, low-order schemata are named *building blocks* and represent partial solutions to

the minimisation problem. A GA discovers new solutions by speculating on many combinations of the best partial solutions contained in the current population.
Since every string is actually representative of many different schemata, an estimate can be done about the order of magnitude of the number of schemata that are processed by a GA population of n strings. It can be shown that a GA processes something like n^3 schemata. For this reason, GAs are said to provide *implicit parallelism*. Basically the simple processing of n strings implies the processing of many more schemata. This gives binary GAs their power of search.

4.8.2.1.2 Limitations and problems of binary GAs

A very large number of modifications have been proposed to enhance the power of search of such a simple method as the one described in the previous section. However, the three-operator binary algorithm represents the backbone of most GA applications. Even though many of them turned out to be successful, these simulations have revealed some drawbacks, which can be regarded as inherent to binary GAs. The main ones are outlined below.

1. *Premature convergence*. The phenomenon occurs when, early in the run, some superindividuals acquire more and more representatives because of their high fitness with respect to the rest of the population. Early convergence to these strings can happen although they relate to a local optimum, which may be well far apart from the global one. Early loss of diversity in the population can be prevented by attenuating the competition among strings through a number of techniques. Dynamic mapping of the objective function and use of diploids are some of the most widespread.
2. *Poor local tuning*. An outstanding advantage of GAs as opposed to typical calculus-based hill-climbing techniques is the global search they can perform. However, a predictable pitfall is the inability to refine the solution once the area of the global minimum has been reached. A straightforward approach would be to use the GA as a pre-processor to perform the initial search, before turning the search process over to a system that can employ domain knowledge to guide the local search. This is due to two main factors:
   - in GAs a sort of hill-climbing is realised through combination of selection and mutation. Nonetheless, its performance is limited when compared to the one provided by specifically designed hill-climbing methods. Local search actually requires the utilisation of higher order and longer defining length than those suggested by the fundamental theorem
   - local tuning is also made difficult by the intrinsic resolution capability of the mapping of the real parameters onto binary strings. This problem is particularly serious when the parameter domains are large, many parameters have to be handled and high precision is required. In this case, the length of the binary solution vector is quite significant. For such problems, the performance of genetic algorithms is quite poor, because the search space is too vast.
3. *Non-trivial constraints*. As many engineering problems are constrained, GA's performance should be tested in presence of constraints. There exist a large number of methods for embedding constraints in a GA-based optimiser. Among them, three main classes of algorithms can be identified (see section 4.9.3). All the methods show

inherent drawbacks, especially when constraints are non-trivial (Michalewicz and Janikow, 1991).

Whereas premature convergence (point 1 above) affects both binary and real-coded GAs in the same way, poor local tuning (point 2 above) is typical of binary GAs. Furthermore, use of real-coded GAs makes solution of problems related to constraints' handling easier.

4.8.2.1.3 Comparison between low and high cardinality alphabets

As mentioned elsewhere, most of the work on GAs is based on binary representation. The main reasons for this wide spread are the following:

- analysis of binary vectors is rather simple
- genetic operators applied to binary strings are simple and even elegant
- a binary alphabet enables maximisation of the number of schemata available for genetic processing. It can be shown (Goldberg, 1989) that the number n_s of schemata per bit of information for strings coded using an alphabet of cardinality k is:

$$n_s = (k+1)^{\frac{1}{\log_2 k}} \tag{4.39}$$

Eq. (4.39) directly results from the fundamental theorem of genetic algorithms. Since function (4.39) is monotonic decreasing, the largest number of schemata per unit information is assured by the binary representation. As GAs derive their power of search just from their implicit parallelism, a binary representation should warrant better performance than any high cardinality representation.

In particular, the claimed capability of binary strings to bear the largest possible number of schemata for the same amount of computer memory used has given this kind of GAs a sound theoretical background.

However, as a matter of fact, the use of high cardinality alphabets as of late is on the rise. Real-coded GAs can be regarded as the ones having the highest cardinality possible. More and more GA applications to complex, real world problems are based on real coding, despite the issue about the number of schemata.
The main advantages of real-coded GAs are the following:

- *comfort with one gene-one variable correspondence*. This point is more psychological than technical. The use of real-coded strings simply allows an easier implementation
- *Hamming cliffs can be avoided.* By means of combined selection and mutation, GAs realise a sort of hill climbing. While real-coded GAs usually adopt mutation operators that perturb the current solution a little about the current value, binary-coded GAs usually adopt a bitwise complement operator for mutation. Thus, while real-coded GAs can easily perform hill climbing in the underlying decision space, binary-coded GAs can become stuck when the fitness function is characterised by the so-called Hamming cliffs. Fig. 4.7 shows a 3-bit function with a Hamming cliff.

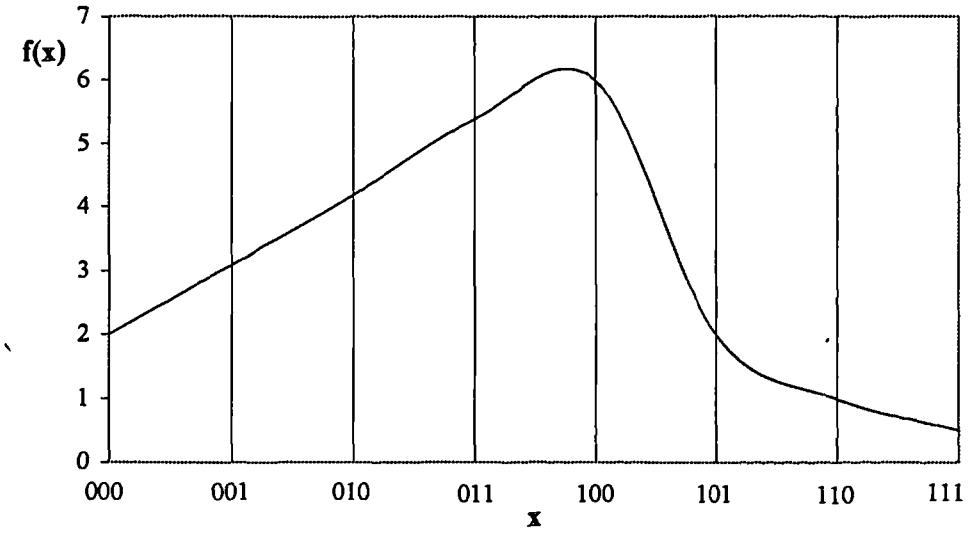


Fig. 4.7: a 3-bit function with a Hamming cliff

Using an octal coding over the eight points of the unimodal search space and some simple kind of mutation, a GA will be able to find the best point regardless of the initial population. After the population converges to some value, successive mutations will continue to correct the solution until the optimum is reached. On the other hand, depending upon initial convergence, the binary GA may or may not be able to access the best point. For instance, because the points in the left half of the space are above average (because $f(0**) > f(1**)$), a GA is fairly likely to converge initially to 011. Although 011 is close to the correct solution as measured in the decision space, it is quite distant in the coded space (also named Hamming space): all 3 bits have to change to reach the optimum 100. Such changes are unlikely and require long waiting times.

- *Speed of convergence*. Higher cardinality alphabets can be shown both theoretically and empirically to converge to the solution more quickly than those coded over a smaller alphabet.
- *Dimensionality reduction*. This in turn reduces the probability of deception and then convergence to incorrect solutions.

On the one hand theoreticians have wondered why practitioners have paid so little attention to their theory, on the other hand practitioners have wondered why the theory seems so unable to come to terms with their findings. The debate between practitioners and theoreticians over this *paradox of real codings* has risen almost to the point of schism.

The problem of the comparison between real and binary coding in GAs can be viewed from two different perspectives. Brief mention is given below, as to how this dichotomy can be compounded.

New interpretation of schema notation. The theory of schemata relies on the "don't care" symbol #. As explained earlier, the character # is used to identify all the strings

which have any gene value at the corresponding position. For example, 000# in binary code identifies both 0000 and 0001. If this simple definition of the "don't care" character is accepted, no quantification over the subset of values at a string position is introduced. Another definition of the character # can substantially modify the way schemata are counted (Antonisse, 1990). The character "don't care" can be defined in order to quantify over the subsets of values at a given string position. It can actually be defined to denote the set of strings sharing a subset of possible values at a position as well as the set of strings of any of the possible values at the position. For simplicity's sake, a language of strings of length 4 and alphabet $\{0,1,2\}$ is used. If the classic definition is accepted, the schema 000# simply refers to the following set of strings: $\{0000;0001;0002\}$. If the new definition is accepted, though, a set of schemata need to be defined to adequately express the quantification over values, because the "don't care" for a 3-valued alphabet can be between 0 and 1, 0 and 2, 1 and 2 or 0, 1 and 2. The quantification over values at a position leads to the following sets: $\{0000;0001\}$, $\{0000;0002\}$, $\{0001;0002\}$ and $\{0000;0001;0002\}$. Therefore, it is necessary to introduce 4 different "don't care" symbols: $\#_{01}$, $\#_{02}$, $\#_{12}$ and $\#_{012}$. Given this new definition, the number N_s of schemata necessary to cover a given string language of cardinality k and length l is:

$$N_s = (2^k - 1)^l \tag{4.40}$$

whereas the number N_s' of schemata sampled by an instance of a string is:

$$N_s' = (2^{k-1})^l \tag{4.41}$$

Thus, if the new definition of the "don't care" character is accepted, a higher cardinality allows processing of many more schemata. This new interpretation of schemata aligns theory with the intuition that a more expressive language should provide better adaptation and power of search.

Virtual alphabets. A theory for convergence of real-coded GAs has been introduced, which is consistent with the classic theory of schemata (Goldberg, 1990). The main outcome of this theory is that selection strongly dominates early GA performance and restricts subsequent search to intervals with above-average function value dimension by dimension. These intervals may be further subdivided on the basis of their attraction under genetic hill climbing (given by the combination of selection and mutation). Each of these subintervals is called a *virtual character* and the collection of characters along a given dimension is called a *virtual alphabet.* During the recombinative phase of the GA convergence, the virtual alphabet is searched and in a large number of problems this is sufficient to ensure that good solutions are found. Although the theory introduced helps suggest why many problems have been successfully solved by means of real-coded GAs, it also suggests that they can be blocked from further progress in those situations when local optima separate the virtual characters from the global optimum. In these cases,

genetic operators different from the usual ones (selection, intergene crossover and mutation) have to be devised to overcome blocking.

4.8.2.1.4 Real-coded GAs

In the engineering community, many different kinds of real-coded GAs have been implemented and tested. In the sequel a real-coded GA, which can be regarded as quite conventional (Janikow and Michalewicz, 1991), is described in order to define the algorithm properly.

- *Representation.* Each string is simply a real (or floating point) vector. Each gene coincides with a decision variable. The precision of this kind of approach depends on the underlying machine, but is generally much better than that provided by the binary representation. In addition, the floating point representation is capable of representing quite large domains. Instead, the binary representation must sacrifice the precision for an increase in domain size, given the fixed binary length.
- *Selection.* It is performed on the basis of the fitness function values in the same way as in binary GAs.
- *Crossover.* Vectors are paired off and a random crossover point is chosen according to a given probability of crossover. The parts after the chosen point are swapped between the two vectors.
- *Mutation.* On the basis of a given probability of mutation, single vector elements are randomly varied inside an assigned range.

The algorithm is analogue to the binary one:

- an initial population is generated randomly
- all strings are assigned their fitness values
- the following steps are repeated:
  - selection is performed, according to a chosen sampling algorithm based on the fitness values (e.g. *roulette wheel selection)*
  - strings are paired off and crossover is applied, according to the probability of crossover
  - strings are subject to mutation, according to the probability of mutation
  - strings' fitness is evaluated

It is worth noting that for the same value of the probability of mutation the simple random mutation described above is somewhat more random than the mutation used for binary strings, where changing a random bit does not imply producing a totally random value from the domain. Therefore, a new operator has been introduced, which performs *dynamic mutation.* If $\mathbf{x}(t)$:

$$\mathbf{x}(t) = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_k \\ . \\ x_n \end{bmatrix} \tag{4.42}$$

is the parameter vector at iteration t and its k-th element is selected for mutation, after that the vector will be:

$$\mathbf{x}(t+1) = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_k' \\ . \\ x_n \end{bmatrix} \tag{4.43}$$

where:

$$x_k' = \begin{cases} x_k + \Delta(t, UB - x_k) & \text{if a random digit is 0} \\ x_k - \Delta(t, x_k - LB) & \text{if a random digit is 1} \end{cases} \tag{4.44}$$

and:

- UB is the upper bound of the range allowed for the considered vector element
- LB is the lower bound
- $\Delta(t,y)$ is a function that returns a value in the range $[0,y]$ such that the probability of $\Delta(t,y)$ being close to 0 increases as t increases. The following function can be used:

$$\Delta(t,y) = y \cdot (1 - r^{\left(1-\frac{t}{T}\right)^b}) \tag{4.45}$$

where:

- r is a uniform distribution random number in the range [0,1]
- T is the maximum number of generations
- b is a constant determining the degree of dependency on the iteration number (e.g. $b = 5$).

The function $\Delta(t, y)$ is shown in fig. 4.8 and 4.9 for two successive values of t. While at the beginning of the run large mutation changes are relatively likely to happen, in the second part of the run large modifications become more and more unlikely. Large changes at an early stage allow the GA to perform a global search, whereas the small changes occurring at later stages localise the search, so that a form of hill climbing can be effected.

The dynamic mutation operator usually warrants much better convergence and enables overcoming of the blocking problem. A comparison with a binary-coded GA (Janikow and Michalewicz, 1991) can be summarised in the following advantages:

- higher speed of convergence
- more consistency from run to run
- higher precision
- ease of implementation
- easy design of operators

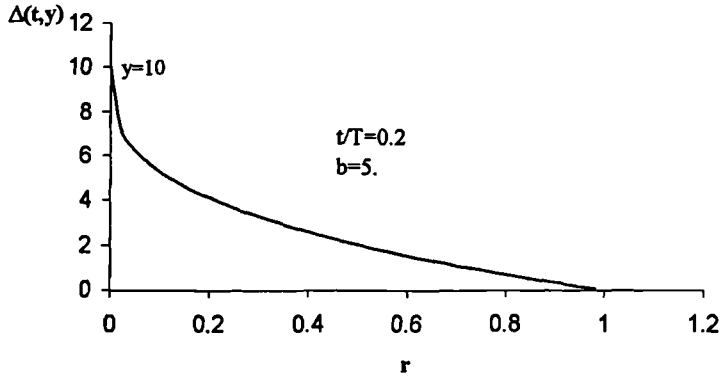


Fig. 4.8: function $\Delta(t, y)$ at an early generation

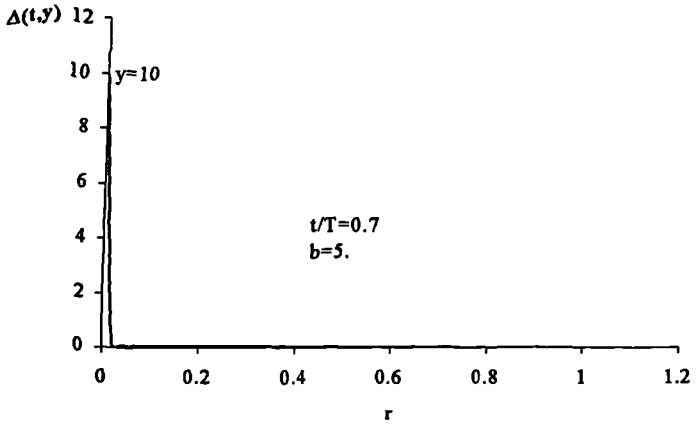


Fig. 4.9: function $\Delta(t, y)$ at a later generation

4.8.2.2 What is an Evolution Program?

The EP is a probabilistic algorithm, used for optimisation, which maintains a population of individuals, $\mathbf{P}(t) = [\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_n^t]$ for each iteration t. Each individual represents a potential solution to the problem at hand and, in any EP, is implemented as some ***data structure*** S. Each solution $\mathbf{x}_i^t$ is evaluated to give some measure of its "fitness". Then, a new population (iteration $t+1$) is formed by selecting the fitter individuals (select step). Some members of the new population undergo transformations (alter step) by means of ***"genetic" operators*** to form new solutions. There are unary transformations (mutation type), which create new individuals by a small change in a single individual, and higher order transformations (crossover type), which create new individuals by combining parts from several individuals. After some number of generations the program converges. The best individual is hoped to represent an optimal or near optimal solution. Fig. 4.10 shows the structure of an EP.

```
program evolution
do
        t ← 0
        initialise P(t)
        evaluate P(t)
        while (not termination condition)
        do
                t ← t+1
                select P(t) from P(t-1)
                alter P(t)
                evaluate P(t)
        enddo
enddo
```

Fig. 4.10: structure of an evolution program

In simple words, all EPs share a common principle: a population of individuals undergoes some transformations and during this evolution process the individuals strive for survival. Binary GAs certainly fit into this definition, as their processing can be represented by the structure shown in fig. 4.11. However, the EP scheme is more comprehensive. Its peculiar features are the following:

- representation: it is tailored to the problem to be solved and takes advantage of specific knowledge
- operators: they are specifically designed for the problem at hand.

The incorporation of problem-specific knowledge in the strings' data structure and the specific operators allows overcoming many problems encountered by binary GAs. Most of the problems affecting binary GAs' performance simply stem from their extreme domain independence. Thus, a key point in the success of binary GAs can also be regarded as the cause of their inherent performance limitation.

In general, Artificial Intelligence problem solving strategies are categorised into strong and weak methods. A weak method makes few assumptions about the problem domain; hence it usually enjoys wide applicability. On the other hand, it can suffer from combinatorially explosive solution costs when scaling up to larger problems. This can be avoided by making strong assumptions about the problem domain and consequently exploiting these assumptions in the problem solving method. But a disadvantage of such strong methods is their limited applicability: very often they require significant redesign when applied even to related problems. EPs fit somewhere between weak and strong methods.

A basic conceptual difference between a classic binary GA and a proper EP approach is shown in fig. 4.11 and 4.12.

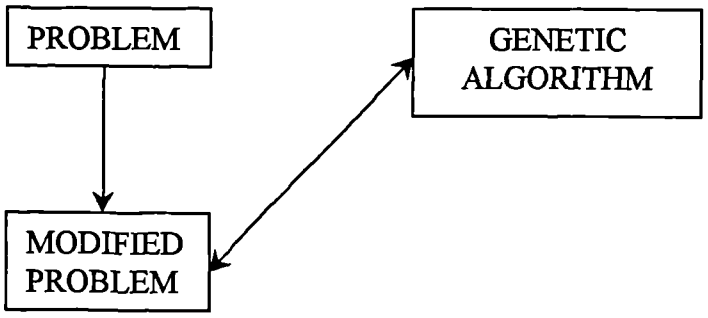


Fig. 4.11: classic GA approach

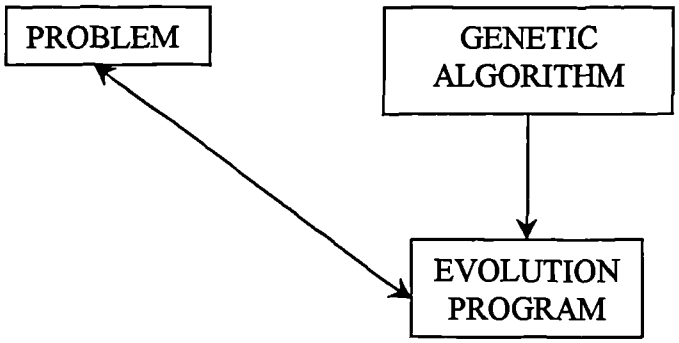


Fig. 4.12: EP approach

The main advantage of EPs is that the problem remains unchanged and is tackled by means of "natural" data structures and domain-specific genetic operators.

Having said all this, it is worth to stress that:

- even a classic binary GA fits in the general scheme represented by fig. 4.10
- it is sometimes quite hard to draw a line between GAs and EPs.

Real-coded GAs may fit in the EP scheme better, as they directly work with problem domain parameters. Enhancement by design of specific operators gets real-coded GAs closer to the typical EP scheme.

An obvious drawback of EP-based optimisers is the poor theoretical basis. In the present work, though, the approach pursued has been a practical one: given a certain objective function, a suitable optimisation method had to be found. Therefore, assessment of the techniques' suitability has been mainly based on the results, although when possible theoretical observations have been made as well.

4.8.2.3 Evolution Strategies

Evolution Strategies (ESs) are optimisation techniques that seem to be suitable for the problem at hand. They can be regarded as evolution programs where a floating point number representation is used, with mutation being the main recombination operator. In the following a brief introduction to ESs is given, from the simple early methods to the refined techniques currently widely utilised.

The earliest ESs are based on a population consisting of a single individual, to which only one genetic operator, i.e. mutation, is applied. The novelty is the representation of the individual as a pair of float-valued vectors:

$$\mathbf{v} = (\mathbf{x}, \boldsymbol{\sigma}) \tag{4.46}$$

where:

- $\mathbf{x}$ is a point in the search space
- $\boldsymbol{\sigma}$ is a vector of standard deviations.

The mutation operator is defined as follows:

$$\mathbf{x}^{t+1} = \mathbf{x}^{t} + N(0, \boldsymbol{\sigma}) \tag{4.47}$$

where:

- $\mathbf{x}^{t}$ is the solution proposed at time t
- $N(0,\boldsymbol{\sigma})$ is a vector of independent random Gaussian numbers with a mean of zero and standard deviations $\boldsymbol{\sigma}$.

The offspring, produced by mutation, is accepted as a new member of the population, i.e. it replaces the parent, if and only if it has better fitness and all constraints (if any) are satisfied. Otherwise, the offspring is eliminated and the population remains unchanged. An useful notation is introduced, according to which the strategy just described is a (1+1)-ES. It means that an initial population made of just one individual is processed to create an offspring. The resulting population is then made of two individuals, among which the best one is selected based on the fitness value.
It is possible to prove a convergence theorem for two-membered ESs, which applies when the optimisation problem is *regular*.

An optimisation problem is regular if the objective function f is continuous, the domain of the function is a closed set, for all $\varepsilon > 0$ the set of all internal points of the domain for which the function differs from the optimal value less than ε is non-empty and for $\mathbf{x}_0$ the set of all points for which the function has values less than or equal to $f(\mathbf{x}_0)$ (for minimisation problems) is a closed set.
If the optimisation problem is regular and all components of the standard deviation vector are identical, the following ***convergence theorem*** can be proved:
for $\sigma > 0$ and a regular optimisation problem with $f_{opt} > -\infty$ (minimisation) or $f_{opt} < +\infty$ (maximisation),

$$p\{\lim_{t\to\infty} f(\mathbf{x}_t) = f_{opt}\} = 1 \tag{4.48}$$

holds.
The convergence theorem states that the global optimum is found with probability one for sufficiently long time. However, it does not provide any clues for the convergence rate, defined as the quotient of the distance covered towards the optimum and the number of elapsed generations needed to cover this distance. Moreover, varying the standard deviations according to the recent convergence performance could help improve the convergence rate.

In an attempt to choose the mutation standard deviations for improved ES convergence, two functions have been analysed (the sphere and the corridor models) and the convergence rate has been maximised with respect to the standard deviation. For the two models, the $1/5$ *success rule* is as follows: the ratio of successful mutations to all mutations should be 1/5. If it is greater than 1/5, the mutation variance must be increased; if it is less, the mutation variance must be decreased. Although in general problems of interest may have characteristics different from those of these two model functions, heuristics have been introduced to adjust the mutation standard deviation along the same line. Every k generations (where k is another parameter of the method) the success ratio of the mutation operator is calculated. Again, if greater than 1/5, it is increased, if smaller than 1/5, it is decreased. The rationale behind the 1/5 rule is that if successful, the search should continue in larger steps, if not, the steps should be smaller. However, it turned out that this search can lead to premature convergence for functions characterised by discontinuities or active constraints.

In order to overcome the problems encountered with the two-membered strategy, the size of the population is increased. If μ is the number of parents that can participate in the generation of one offspring individual, two of them will be picked up to produce an offspring by crossover. The offspring will be modified by means of Gaussian mutation and then the least fit individual in the $\mu+1$ population is discarded. This strategy is denoted as $(\mu+1)$-ES.

A further modification that warrants better performance is to allow control parameters, such as the mutation variance, to self-adapt rather than let them change according to

some deterministic algorithm (e.g. the 1/5 success rule). Therefore, the $(\mu+\lambda)$-ES is introduced. This is a natural extension of the $(\mu+1)$-ES, where μ individuals produce λ offspring by means of crossover and selection. The new, temporary population of $(\mu+\lambda)$ individuals is reduced by a selection process again to μ individuals. The mutation operator applied to an individual defined by $(\mathbf{x},\boldsymbol{\sigma})$ generates a new individual $(\mathbf{x}',\boldsymbol{\sigma}')$ according to the following rules:

$$\boldsymbol{\sigma}' = \boldsymbol{\sigma} \cdot e^{N(0,\Delta\sigma)} \tag{4.49}$$

$$\mathbf{x}' = \mathbf{x} + N(0,\boldsymbol{\sigma}') \tag{4.50}$$

where $\Delta\sigma$ is a parameter of the method.
The basic idea is that individuals with better-adjusted strategy parameters $\boldsymbol{\sigma}$'s are expected to perform better and selection will favour them. Sooner or later, then, those individuals will dominate the population. As the notation suggests, in any $(\mu+\lambda)$-ES selection operates on the joint set of parents and offspring. Thus, parents will survive until they are superseded by better offspring. It might even be possible for very well adapted individuals to survive forever. This feature gives rise to some deficiency of the $(\mu+\lambda)$-ES:

- on problems with an optimum moving over time a $(\mu+\lambda)$-ES gets stuck at an out-dated good location if the internal parameter setting becomes unsuitable to jump to the new field of possible improvements
- the same happens if the measurement of the fitness (hence the objective function) is subject to noise
- it can be shown that if μ/λ is greater or equal to the probability for a successful mutation, there is a deterministic selection advantage for those offspring which reduce some of their σ_i's.

A straightforward modification of the $(\mu+\lambda)$-ES, which allows overcoming the drawbacks mentioned above, is the (μ,λ)-ES. In this case, μ parents generate λ offspring and selection is applied just to the offspring. Thus, the lifetime of every individual is restricted to one generation. The limited life span allows forgetting inappropriate internal parameter settings. This may result in short phases of recession, but avoids long stagnation phases due to misadapted strategy parameters.

A final refinement is based on the introduction of an additional control parameter, $\boldsymbol{\theta}$, which is used to define the direction of the mutation changes. The individual is then represented by $(\mathbf{x},\boldsymbol{\sigma},\boldsymbol{\theta})$. In ESs mutation realises a kind of hill climbing search, when considered in combination with selection. If each variable x_i has got its own σ_i, preferred directions of search can be established only along the axes of the co-ordinate system. In general, the best search direction is not aligned with those axes (see also section 4.8.1) and thus the trajectory of the population through the search space zigzags trying to reach the bottom of the valley (or the top of the mountain in case of maximisation). In order to avoid the consequent reduction of the rate of progress, an

extended mutation operator is introduced to handle *correlated mutations* completely defined by addition of the control parameter $\boldsymbol{\theta}$. The corresponding mutation operator is:

$$\boldsymbol{\sigma}' = \boldsymbol{\sigma} \cdot e^{N(0,\Delta\sigma)} \tag{4.51}$$

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \mathbf{N}_0(\Delta\theta) \tag{4.52}$$

$$\mathbf{x}' = \mathbf{x} + \mathbf{C}_0(\boldsymbol{\sigma}', \boldsymbol{\theta}') \tag{4.53}$$

where $\mathbf{C}_0(\boldsymbol{\sigma}', \boldsymbol{\theta}')$ is a normally distributed random vector with expectation zero and probability density defined as follows:

$$p(\mathbf{C}_0(\boldsymbol{\sigma}', \boldsymbol{\theta}')) = \sqrt{\frac{\det A}{(2\pi)^n}} \cdot e^{\left[-\frac{1}{2}\mathbf{N}_0(\sigma')^T \cdot A \cdot \mathbf{N}_0(\sigma')\right]} \tag{4.54}$$

The diagonal elements of the covariance matrix A^{-1} are the independent variances $\sigma_i'^2$, whilst the off-diagonal elements represent the covariances c_{ij} of the mutations. The space of equal probability density is restricted to the surface of the n-th dimensional rotating hyperellipsoids that are realised by a set of inclination angles $\boldsymbol{\theta}'$ of the main axes of the hyperellipsoid. The number of such parameters is $n(n-1)/2$.
The introduction of the additional strategy parameters $\boldsymbol{\theta}$ allow for correlated mutations and thus self-learning of effective trajectories in the topological environment.

4.8.2.4 Comparison of GAs and ESs

Both GAs and ESs can be considered evolution programs. In particular, both systems maintain populations of potential solutions and make use of the selection principle of the survival of the fitter individuals. However, there exist many differences between these two approaches. Some of them are outlined below.

- *Domain of application.* ESs were explicitly developed as methods for numerical optimisation. They adopt a special hill climbing procedure with self-adapting step sizes and inclination angles. On the other hand, GAs were initially meant to be general-purpose adaptive search techniques, which allocate exponentially increasing number of trials for above-average schemata. GAs have been applied in a variety of domains and real parameter optimisation was just one field of application.
- *Representation*: ESs always operate on floating point vectors, whereas classic GAs operate on binary vectors.
- *Selection*: the ES selection mechanism is deterministic, the GA one is probabilistic.
- *Order of operators*: in ESs, the selection process follows application of recombination operators (crossover and mutation), whereas in GAs these steps occur in the opposite order.
- *Control parameters.* Typically, they are constant in GAs and variable in ESs.

- *Handling constraints.* Given inequalities constraints, if during an iteration of an ES an offspring does not satisfy all of them, then the offspring is disqualified, i.e. it is not placed in the new population (*barrier method*). If the rate of occurrence of such illegal offspring is high, the ES adjusts its control parameters by decreasing the components of vector $\boldsymbol{\sigma}$. On the other hand, the major strategy for GAs to handle constraints is to impose penalties on individuals that violate them. As stated earlier (see section 4.8.1), the penalty function technique has many disadvantages, one of which is problem dependence. The issues related to application of GAs to constrained optimisation problems are analysed in section 4.9.3.

The GA and ES communities, although separate, have influenced each other over the years. Many concepts have been exchanged between them. Clearly, the GA community has borrowed the idea of real vector representation from ESs (currently a large number of GA applications are based on real coding) and the ES community has borrowed the crossover operator from the GA community.

4.9 Development of the Evolution Program for fault diagnosis

As stated in section 4.8.2, conventional, calculus-based methods turned out to be insufficient to accomplish the constrained optimisation task defined in section 4.7.3. Therefore, evolutionary optimisation techniques have been implemented and tested in various forms.

4.9.1 A simple binary GA

At first, a standard binary GA, reliant on the three basic operators (selection, crossover and mutation), was implemented and used to optimise function (4.25)-(4.26) without accounting for the constraint on the maximum number of fault-affected performance parameters. This choice was made to test the overall performance of the GA in a relatively simple task.
Usually a large population, made of a number of strings ranging from 200 to 1000, was used. The probability of crossover p_m was set to 0.6, the probability of mutation p_c was set to 0.01. Apart from the smearing that, as expected, affected the results, the following remarks can be made:

- although GAs can be regarded as inefficient from the point of view of the number of objective function's evaluations and the performance simulation code used is detailed and therefore rather heavy computationally, convergence could be reached by using reasonable computing resources (i.e. memory and time)
- often the binary GA converged to a solution completely different from the correct one. This might be due either to the actual presence of local minima or to the GA inability to optimise the objective function at hand. Hamming cliffs could be one of the causes. As a matter of fact, the lowest value of the objective function obtained during the run of the GA was usually far from the value corresponding to the correct solution

- the GA seemed to face problems in the choice of the subset of biased measurements to be excluded from the evaluation of the objective function. When early in the run the fittest strings were characterised by a wrong subset of biased measurements, the GA found it difficult to jump to another subset as a consequence of the optimisation. This lack of flexibility in performing the sensor validation task was similar, even though slightly better, to the one shown by calculus-based methods (section 4.8.1)
- slightly higher rate of convergence was obtained by substituting squares for the absolute values, as required by (4.25). In this case convergence to a solution was usually reached after fewer iterations but the accuracy was still low.

In the light of the points made above, the overall performance of the standard binary GA-based optimiser was deemed unsatisfactory. However, it is worth pointing out that the outcomes of this analysis are to be considered partial, in that application of the constraint on the maximum number of fault-affected performance parameters definitely changes the optimisation problem substantially.
In the quest for improvements to be applied to the standard GA described above, two modifications, which are described below, turned out to be effective.

Mapping. Engineering optimisation problems require maximisation or minimisation of an objective function, which has some direct physical meaning. Before selection, GAs require a fitness function to be assigned to each string as a figure of merit. The simplest approach is to choose the fitness function coincident with the objective function. However, the fitness function usually differs from the objective function for a number of reasons. First of all, because of the way selection algorithms work, the fitness function must be positive. This is not always the case for the objective function, which may be positive or negative depending on the value of the variables. Furthermore, the GA maximises the fitness function. That means that an objective function to be minimised has to be mapped onto a fitness function to be maximised. Thus, often mapping is required for the simple reasons explained. Moreover, careful mapping can be exploited to help GA convergence. A common choice for the mapping is the linear function:

$$F = a \cdot J + b \tag{4.55}$$

where:

- J is the objective function
- F is the fitness function
- a and b are constant.

Actually a and b can be adequately adapted to prevent or limit GA's premature convergence. As described in section 4.8.2.1.2, premature convergence is a common problem for GAs. It occurs when, early in the run, quite fit individuals quickly dominate the population, although they are far from the correct solution. At this stage, attenuation of the competition among strings would be beneficial, as diversity would be preserved and potentially useful substrings could be saved and exploited in subsequent generations. On the other hand later in the run the mean and the maximum fitnesses turn out to be rather close. At this stage, increasing the competition can foster convergence to the correct solution. This strategy can be implemented by imposing two adequate conditions to calculate a and b (Goldberg, 1989). At the beginning these are: the mean fitness is equal to the mean objective function and the maximum fitness is equal to a constant (c)

times the mean fitness. This strategy allows to scale down the strings' fitness early in the run and to limit the premature convergence. Later in the run, on the contrary, competition is increased by imposing the same condition on the mean values as before, whereas the second condition is that the minimum objective function corresponds to a zero-valued fitness function. The algorithm just described has been introduced by Goldberg (1989) and can be applied when the original optimisation is already a maximisation. When the objective function has to be minimised, the concepts to be applied are similar.

At first, a very simple form of linear mapping was used: $a=-1$ and $b=J_{sup}$ where chosen, where J_{sup} is a value known to be greater than the maximum J to be expected. This mapping, though, does not address the issue of premature convergence. However, application of linear mapping did not allow reducing the inaccuracy.

On the contrary, a significant improvement was obtained by application of a non-linear mapping (Chen et al., 1994). In this case the relationship between fitness and objective function is defined as follows:

$$F=\frac{1}{J} \tag{4.56}$$

The hyperbolic relationship between F and J implies that when a large objective function is calculated the corresponding fitness is very small and when a small objective function is calculated the corresponding fitness is very large. The improvement in accuracy that was obtained can be explained as a need to "localise" the search. The typical lack of local search capability of GAs seemed to be a major cause for the convergence failures. This was attenuated by means of non-linear mapping.

Sampling algorithm. The selection phase determines the actual number of offspring each individual will receive based on its relative performance. The selection phase is made of two parts: 1) determination of the individuals' expected values (see (4.37)) 2) conversion of the expected values to a discrete number of offspring. The algorithm used to convert the real expected values to integer numbers of offspring is called the sampling algorithm. The sampling algorithm must maintain a constant population size while attempting to provide accurate, consistent and efficient sampling. Three key points define the performance of a sampling algorithm:

1) Bias: it is defined as the absolute difference between an individual's actual sampling probability and his expected value. The expected values should be sampled as accurately as possible. The optimal, zero bias is achieved whenever each individual's sampling probability equals his expected value.
2) Spread: it is the range of possible values for the actual number of offspring a given individual receives in a given generation. Whereas the bias indicates accuracy, the spread indicates precision. Hence the spread reveals the algorithm's consistency.
3) Efficiency: it is a measure of the computing resources used for sampling. Obviously, it is desirable for the sampling not to increase the GA's overall time complexity.

The most used sampling algorithm is the *roulette wheel*, described below:

- determine R, the sum of the competing expected values

- map the individuals to contiguous segments of the real number line, $[0;R]$, such that each individual's segment is equal in size to its competing expected value
- generate a random number within $[0;R]$
- select the individual whose segment spans the random number
- repeat the process until the desired number of samples is obtained.

The algorithm's name stems from the analogy with a gambler's spinning wheel where each wheel slice is proportional in size to some individual's expected value. In terms of computing time, this technique is $O(N^2)$ (i.e. is of the order of the square of the number of strings N making up the population).

Large bias, large spread and low efficiency affect the roulette wheel-sampling algorithm. Therefore, several modifications have been introduced to improve its performance. Among them, the most widespread is the *remainder stochastic sampling without replacement*. In a first phase, also named integral, samples are awarded deterministically on the basis on the expected values. In the second phase, also named fractional, a spinning wheel is used to sample according to the expected values' fractional portions. However, after each spin, the selected individual's expected value is set to zero. Hence, individuals are prevented from having multiple selections during the fractional phase. Although very used, the algorithm is biased toward smaller fractions. In terms of the three key points explained above, the sampling algorithm shows low bias, minimum spread and a computational cost equal to $O(N)$.

In the present work, both the roulette wheel and the remainder stochastic sampling without replacement where tested. Even though a certain improvement was observed, the GA's performance did not benefit in a significant way.

Another sampling algorithm, the *Stochastic Universal Sampling* (*SUS*), has been implemented and tested, as it is claimed to be better than the previous ones in terms of bias (Baker, 1987). On a standard spinning wheel, there is a single pointer that indicates the winner. The SUS algorithm is analogous to a spinning wheel with N equally spaced pointers. Hence, a single spin results in N winners. This sampling algorithm is zero bias, minimum spread and has a computational cost equal to $O(N)$. Therefore, it seems to be able to provide better performance than the other two sampling mechanisms just described. Tests carried out for the problem at hand confirmed that the convergence substantially benefited from application of the SUS algorithm.

Eventually, it is worth to discuss the application of elitist techniques briefly.

Elitism. Although the sampling algorithm should select the fitter individuals to be part of the next generation, sampling inaccuracy can prevent the current best individual from being selected. In order to preserve the current best string, the selection algorithm can be modified in order to include it in the next generation whatever the choice of the sampling algorithm may be. In the present work, the current best string, when not directly selected, was included in the next generation by substitution for the current worst string. Tests have been carried out to ascertain the effect of elitism on the convergence for the problem at hand. The GA's performance did not seem to benefit from the application of elitism. As pointed out by Goldberg (1989), elitism improves local search but the global search capability is worsened. Although the studies made on mapping seem to suggest that the local search capability had to be boosted, the global search capability still

remains a primary requirement for the optimisation of the objective function. Apparently, a good trade-off has to be found between global and local search capability.

4.9.2 Real coding

Analysis of the results obtained with the binary GA described in section 4.9.1 and the findings described in section 4.8.2.1.4 suggested that a real-coded GA might perform better. Therefore, a real-coded GA was implemented and tested, at first without imposing the constraint on the maximum number of fault-affected performance parameters.
Mapping was non-linear, according to (4.56), selection was performed by using the SUS algorithm, crossover was standard and mutation was dynamic (see section 4.8.2.1.4). Crossover and mutation were applied to the two vectors $\mathbf{x}$ and $\mathbf{w}$ separately. The probability of crossover p_m was set to 0.5, the probability of mutation p_c was set to 0.2 for both $\mathbf{x}$ and $\mathbf{w}$. 1000 strings processed for 200 generation were sufficient to reach convergence.
The real-coded GA, as expected, performed better than the binary one. In particular the following gains were achieved:

- ease of implementation: the data structure used was simply made of a pair of real vectors, $(\mathbf{x},\mathbf{w})$
- higher speed of convergence: the GA converged to a solution close to the actual one by means of fewer generations
- more consistency from run to run: the spread of the estimation errors from run to run was lower with respect to the one provided by the binary GA
- higher precision: the data structure was implemented as a pair of double precision vectors. The choice of using double precision quantities depends on the need to calculate the objective function accurately, with minimal propagation of numerical errors. It is worthwhile to highlight that calculation of the terms to be summed up in the objective function is rather prone to numerical errors, especially at later generations when measured and predicted values are quite close to each other.

However, the smearing effect still affected the results.

4.9.3 Constrained optimisation

As explained in section 4.6 and 4.7.3, an effective way to reduce the smearing should be the application of a constraint on the number of faulty engine components. The assumption is acceptable from the point of view of fault diagnosis and should be helpful from an analytical point of view. For an engine like the EJ200 (and for most engines as well), faults are likely to simultaneously affect one or at most two components. Since the rotating components' performance is usually quantified by means of two performance parameters (typically efficiency and flow function), only four parameters should be simultaneously fault-affected. As the propelling nozzle's performance is expressed by one parameter, the discharge coefficient, and low by-pass ratio turbofans are modelled by using a single overall mass flow function, the number of fault-affected performance

parameters ranges from 1 to 4. It is worth noticing that the actual constraint is on the maximum number of faulty engine components and not fault-affected performance parameters. The latter derives from the former. By stating that a maximum number of four performance parameters are allowed to be simultaneously fault-affected, it is meant that their distribution is such that only two engine components are faulty.
The rationale behind application of this constraint is simple: since a solution is searched for, which should be related to one or two faulty engine components, the optimiser is only allowed to produce strings satisfying this requirement.
Actually the path towards a proper definition of the constraint has gone through guided initialisation first. As a calculus-based iterative technique for solving a non-linear optimisation problem is more likely to converge successfully if the initial guess it starts with is sufficiently close to the solution being sought, at first the population of strings was initialised according to the constraint on the maximum number of fault-affected performance parameters without the constraint being applied during running of the GA. Two remarks have to be made on the behavior of the GA:

- the current best strings had a relatively low value of the objective function at the beginning, compared with the ones obtained with unsupervised initialisation
- the advantage of starting from a good set of points was soon spoilt due to the generation of strings representing faults affecting more than two engine components. As the optimiser kept on cycling from generation to generation, the smearing effect started to appear and strongly affect the accuracy.

Therefore, both supervised initialisation and application of the constraint seemed to be required.

It is here necessary to distinguish between two different types of constraints that have to be met:

- range constraint: sensible assumptions can be made as to the range of variation of the performance parameters $\mathbf{x}$. For the EJ200 in development test bed, a limit of 3% variation is acceptable. As far as the environment and power setting parameters are concerned, there is no similar way to set sensible limits to the variation of the related biases. However, the limits can be either set to a certain value, established by experience, or set to a reasonable large value. In the present work, the range of variation of the i -th environment and power setting parameters has been set to $\pm 60 \cdot \sigma_{wi}$, where σ_{wi} is the i -th measurement noise standard deviation
- constraint of the number of faulty components: its application is focused on reduction of the smearing.

As detailed in section 4.8.1, calculus-based optimisation techniques show to find it difficult to satisfy even range constraints. In that section, the typical difficulties encountered by application of penalty function methods to calculus-based techniques have been briefly discussed. On the contrary, range constraints are implicitly satisfied whenever an EP-based optimiser is used as the range of variation of the variables is set at the beginning, at design phase. In particular initialisation and mutation of the strings have to be provided with upper and lower bounds. Once they are given, all strings that are generated will automatically be part of the allowed space.

Satisfaction of the second constraint, though, is less straightforward. In the following, emphasis is put on the quest for an optimisation technique able to satisfy this constraint. A brief analysis of the three main ways GAs can deal with constraints is given below (Michalewicz and Janikow, 1991; Michalewicz, 1996).

Penalty function. Concepts introduced in section 4.8.1 are here expanded. A minimisation problem is here assumed for convenience. With this constrained opttimisation method potential solutions are generated without considering the constraints and then they are penalised by increasing the objective function by a certain amount. In other words, a constrained problem is transformed to an unconstrained one by associating a penalty with all constraint violations. A simple penalty method is defined by (4.35), where all violations are weighted by the same amount. More complicated weightings can be used. In general, there exists no accepted methodology to combine the penalty term with the original objective function. If a high penalty is incorporated into the evaluation routine and the domain is one in which production of an individual violating the constraint is likely, the GA runs the risk of spending most of the time evaluating illegal individuals. Furthermore, it can happen that when a legal individual is found, it drives the others out and the population converges on it without finding better individuals, since the likely paths to other legal individuals require the production of illegal individuals as intermediate structures and the penalties for violating the constraint make it unlikely that such intermediate structure will reproduce. If moderate penalties are imposed, on the contrary, the system may evolve individuals that violate the constraint but are rated better than those which do not because the rest of the evaluation function can be satisfied better by accepting the moderate constraint penalty than by avoiding it. In conclusion, techniques based on penalty functions usually work reasonably well for narrow classes of problems and for few constraints.

Decoders and repair algorithms. Special representation mappings, named decoders, can be used to guarantee (or at least increase the probability of) the generation of a feasible solution. They are used along with repair algorithms that correct any unfeasible solution. Typical drawbacks of this approach are that the algorithms are frequently computationally intensive to run and not all constraints can be easily implemented in this way.

Specialised data structures and genetic operators. This approach is based on development of a problem-specific EP using appropriate data structures together with a suitable family of applicable genetic operators that can hide the constraints. Drawbacks of this approach are that it is not always possible, for an arbitrary set of constraints, to develop an efficient data structure hiding such constraints. The same applies to the development of the related genetic operators.

In the present work, penalty function techniques and tailored EPs have been implemented and tested. The method based on a problem-specific EP turned out to be much more effective. In the following both attempts are described.

Various types of penalty function methods have been tested. Either high or moderate penalty terms have been added to the objective function depending on the number of

violated constraints (see (4.35)). The remarks to be made about the application of the penalty function methods are:

- faults affecting a single engine component are not easily dealt with, as well as faults affecting a pair of components one of which is the propelling nozzle, whose health is quantified only by one performance parameter. If N_{viol} was set equal to the number of fault-affected parameters minus four (number of performance parameters that are fault-affected when two engine components are faulty), after few generations there was no string representing single component faults anymore. The same problem occurred when the propelling nozzle was faulty together with another component
- the choice of the value L to be used in (4.35) is not easy. It can be considered as a mattèr of tuning and the typical problems claimed to affect the method have actually been encountered. If high penalties were used, the GA frequently found it difficult to move from a solution to another one located in a different area of the space. Basically, the GA got stuck to incorrect solutions. If moderate penalties were imposed, convergence was smoother but the smearing effect was reduced only slightly because of partial application of the constraints
- when high penalties were imposed and the string that at the beginning showed the best fitness was sufficiently close to the correct solution, the smearing effect resulted to be dramatically reduced. As a consequence, the diagnostic answer's accuracy improved significantly.

The tests carried out have shown that:

- the penalty method is usually unable to correctly impose the constraint, due to its inherent features
- when the constraint is successfully imposed, the accuracy improves very much.

The key idea behind the EP approach is as follows: the correct solution is characterised by faults located in just one or at most two engine components. With the supervised initialisation, all strings are legal at the beginning. Actually the population is made of a number of ***fault classes***, each one referring to a certain location of faults. There will be a fault class for faults located in the fan outer, another for faults in the fan inner, another for faults in the HP compressor, etc. Moreover there will be classes for faults located in two components (fan outer-fan inner, fan outer HP compressor, HP turbine-LP turbine, etc.). In general, if N_{comp} is the number of engine components and 2 is the maximum number of components allowed to be simultaneously faulty, the number of fault classes N_{class} can be computed as follows:

$$N_{class} = N_{comp} + \binom{N_{comp}}{2} \tag{4.57}$$

where the first term relates to single faulty components, the second term relates to faults located in two components. For the EJ200, $N_{comp} = 6$, hence $N_{class} = 21$.

After the supervised initialisation, the genetic operators are designed to keep fault classes separate. The GA is based on the three basic operators: selection, crossover and mutation. As crossover and mutation are the operators that can disrupt the fault class scheme, they are applied to the fault classes separately. Therefore, a string of a certain

fault class will be subject to crossover only with strings of the same class. Similarly, mutation will be applied only to the fault-affected performance parameters defining the fault class. As far as selection is concerned, two options are available:

- again, selection is applied to the various fault classes separately. In this way, though, the GA is actually made of N_{class} non-interfering sub-GAs. They are run separately and the one providing the lowest objective function identifies the faulty engine components and thus gives the diagnostic answer
- selection is applied to the entire population. In this case, the fault classes compete with each other to gain more individuals every generation and its application to the population as a whole does not cause mixing of different fault classes.

The second avenue has been pursued and, as shown in the sections on results, the performance of the EP is very good. The first approach has been discarded in favour of the second one because with the former method N_{class} optimisations have to be completed with the initial number of strings assigned to the corresponding fault class. With the latter method, on the contrary, early in the run the GA will concentrate on few fault classes, which will gain more and more individuals. In this way, after few generations uninteresting fault classes are soon discarded and the computational resources are concentrated on the promising fault classes. In simple words, selection extended to the entire population fosters competition among fault classes and so concentration on the faulty engine components, whereas crossover and mutation gradually refine the solution.

An important issue is the distribution of strings among the various fault classes. As a matter of fact, the fault classes are different for a number of reasons. First of all, some of them refer to faults in single components and the others to faults in two components. Moreover, different choices of faulty components can give different numbers of fault-affected performance parameters (N_{perf}). A sensible approach to split the total number of strings among the fault classes would be to assign to each fault class a number of strings proportional to the number of fault-affected performance parameters. However, two remarks have to be made: firstly, the simple proportionality does not take into account that both single and dual fault classes require estimation of the environment and power setting parameters; secondly, and more importantly, tests have shown that this rule of assignment of strings according to the number of fault-affected performance parameters penalises the single fault classes. In fact, if this simple rule is used, in case of faults affecting a single component the diagnostic answer may be wrong. An increase in the number of single fault class strings with respect to the dual fault class strings allows to balance out the situation. For the analysed engines, the ratio was set to 3. Although the exact value of this proportionality constant has been established by trial and error and *the diagnostic accuracy is not a strong function of its value*, understanding its meaning is important. The number initially assigned to a certain fault class could be interpreted as assignment of the a priori probability density function for that fault class (function $p(\mathbf{x})$ of (2.130)). Increasing the number of strings for a certain fault class means making the mapping of the corresponding area finer. Therefore, no matter what the measurements suggest, this fault class is more likely to produce a well-performing string than before the increase.

4.10 Testing with the EJ200

Tests have been made by using an accurate Rolls-Royce steady state non-linear model of the EJ200 (see section 4.4), supposed to be in development test bed (Zedda and Singh, 1999a). As detailed in section 4.3, 10 performance parameters have to be estimated by 13 monitoring measurements. 3 measurements are used to set the operating point of the engine. Data for testing have been obtained by using the performance simulation code in synthesis mode and then superimposing noise and biases.

As detailed in section 4.6, 2 is the maximum number of engine components supposed to be simultaneously faulty. The maximum level of deterioration is set to 3%.

Real noise levels were superimposed to the clean measurements provided by the simulation code. Due to confidentiality issues, they cannot be reported here. However, measurement noise standard deviations are comparable with data available in public literature (Walsh and Fletcher, 1998). It is worth to underline that the noise standard deviations can be substantially different from one another. For instance, spool speed measurements are remarkably more accurate than airflow measurements.
2 or even 4 monitoring measurements are allowed to be biased as well as the 3 environment and power setting parameters. HP and LP spool speed measurements were not assumed to be biased. As a matter of fact, the corresponding sensors are quite reliable. Moreover, occurrence of a sensor fault would clearly manifest itself (no slow drift is likely to occur), so that the probability of regarding a faulty sensor as fault-free is really negligible in this case. This fact is very fortunate due to the great influence of spool speed measurements on engine performance and to the relatively small noise level by which they are affected.
Such a large number of biases as the one assumed is actually highly unlikely to affect an instrumentation set. However, the decision to implant so many biases was made in an attempt to test the method's tolerance for bias. The biases superimposed to monitoring measurements have been chosen small on purpose, because small values are particularly undesirable in that their detection is difficult. The magnitude of the biases has been set to 1% for all measurements, apart from those whose assumed measurement non-repeatability range, equal to $3*\sigma$, is comparable to 1% variation. For these, a level of 2% has been chosen. The biases affecting the environment and power setting measurements have been set to the large value of $50*\sigma$ as accommodation is automatically carried out.

The GA population has been set to 4000 strings and 200 iterations were sufficient to reach convergence. The convergence was monitored by checking on the population diversity. The number of non-empty fault classes is a good measure of convergence during the initial phase of the run. Moreover, during the entire run the performance parameters' standard deviations of the population are monitored. Convergence is reached when the whole population is made of just one fault class and the corresponding standard deviations are sufficiently small.

Fig. 4.13 shows the comparison between typical results provided by the proposed diagnostic system and a straightforward maximum likelihood-based optimisation

minimising function (4.9) with allowance for noise and biases in **w** without constraint on the number of fault-affected parameters. In the former case, 4 biases affected the monitoring measurements, in the latter no bias was present.

parameters	actual (%)	predicted (%)	maximum likelihood (%)
Δηfanout	-3	-2.99	-2.9
ΔΓfan	0	0	-0.27
Δηfanin	0	0	-0.33
ΔΓhpc	3	2.99	2.91
Δηhpc	-1	-1.03	-0.41
ΔΓhpt	0	0	-0.56
Δηhpt	0	0	-0.11
ΔΓlpt	0	0	-0.72
Δηlpt	0	0	-0.2
ΔCa	0	0	-0.27
setting parameters			
Wf	811.728	811.603	811.231
P1	83.688	83.773	83.574
T1	312.02	312.094	311.89
		RMS=0.01	RMS=0.39

Fig. 4.13: comparison in a 2 faulty component test case

The first column ("actual") lists the faults implanted. The faulty components were fan outer and HP compressor. The second column ("predicted") displays the prediction of the proposed diagnostic method. The third column ("maximum likelihood") shows the results provided by the maximum likelihood-based method. The 3 environment and power setting parameters were biased. The performance parameters' estimation error is quantified by the Root Mean Square error (RMS), whose definition is here reported for completeness:

$$RMS = \sqrt{\frac{\sum_{j=1}^{N}(x_j - x_{estj})^2}{N}} \tag{4.58}$$

where:

- N is the number of performance parameters (here $N = 10$)
- x_j is the j-th component of the actual performance parameter vector
- x_{estj} is the j-th component of the estimated performance parameter vector.

The following remarks are made:

- the proposed method allows to clearly identifying the faulty engine components. Smearing is strongly reduced and concentration is accomplished
- the accuracy, expressed by the RMS, is significantly better, even though the maximum likelihood-based optimiser could rely on 4 measurements more
- both techniques are able to accommodate the biased environment and power setting parameters. Thus, the way these biases are dealt with can be considered effective.

Fig. 4.14 shows the values of all the terms that could be used to sum up in the objective function (4.25) in the case of faulty outer fan and HP compressor introduced above. In the considered case the WFE, P13, T13 and P21 are biased. Correctly, the GA identified the first 4 measurements as faulty and then the corresponding terms have not been included in the objective function.

	actual	predicted
W1a	6.59	6.65
P13	13.07	13.28
T13	11.19	11.38
P21	11.84	12.09
T21	0.16	0.02
W21	0.06	0.13
P3	0.73	0.48
T3	0.1	0.19
T5	0.12	0.16
P5	0.81	1.02
F	0.63	0.55
Nhp	0.92	0.77

Fig. 4.14: bias isolation

Accommodation of the 4 biased monitoring measurements is straightforward once the performance parameters have been calculated. A simple run of the performance simulation code in synthesis mode allows calculating the corrected measurements, if necessary.

150 test cases have been run and the average results are summarised in table 4.2. Some samples are shown in appendix J.

RMS is the average value. e_1, e_2 and e_3 are the average errors of estimation of the ambient and power setting parameters, successful component classification (s.c.c.) and successful sensor classification (s.s.c.) are the percentage of correct identification of faulty engine components and sensors respectively.

	2 biases	4 biases
1 faulty component		
RMS	0.037	0.106
e_1 (g/s)	0.44	0.87
e_2 (kN/m^2)	0.11	0.13
e_3 (K)	0.18	0.23
s.c.c. (%)	93.2	95.2
s.s.c. (%)	98.3	97.6
2 faulty components		
RMS	0.181	0.272
e_1 (g/s)	0.86	1.46
e_2 (kN/m^2)	0.22	0.27
e_3 (K)	0.32	0.38
s.c.c. (%)	98.1	96.3
s.s.c. (%)	91.9	91.7

Table 4.2: test case results

The tests made allow drawing the following conclusions, which are to be regarded as statistically significant due to the large number of simulations carried out:

- the estimation accuracy is high, especially when data are analysed in the light of the overall number of biases (5 and 7 respectively out of 16 measurements).
- as expected, an increase in the number of biases produces larger estimation errors in terms of RMS and e_1, e_2 and e_3
- an increase in the number of faulty components also produces larger estimation errors. It is reminded that a larger value of $(N_{perf}+P)/(M-M_{bias})$ implies less relative redundancy
- conversely, the capability to isolate faulty engine components and sensors does not generally seem to be dependent on the number of biases
- the percentages of successful component classification show how much the "smearing" effect has been reduced. The better s.c.c. performance provided in the two faulty component cases is due to a number of runs where a single actually faulty component has been identified, but a small deterioration has wrongly been found in the nozzle as well. Usually these errors affect the estimation RMS only marginally.

It is worthwhile to point out that although any fault relating to a single faulty component can be represented by a string of a two component fault class provided that the fault free component's performance parameters are zeros, this occurrence is very rare except for the nozzle.

Whereas the GA automatically identifies during the convergence whether one or two engine components are faulty, the number of supposedly biased measurements has to be set since the beginning.
If the actual biased measurements are more than the assumed ones (M_{bias}), then the optimiser is likely to converge to an incorrect solution due to the effect of the undetected

biases. On the contrary, if M_{bias} is larger than the number of actual biases, the optimiser is likely to estimate the engine faults correctly, because the biased measurements will be isolated together with other fault-free sensors in order to get the most consistent solution. M_{bias} should be chosen on the basis of specific knowledge about the typical occurrence of sensor faults for the given engine-sensor suite. In the considered case of test bed for a two spool development engine, occurrence of faults in more than two sensors has to be regarded as unlikely. However, the choice of M_{bias} is usually not crucial, because the number can safely be chosen to overestimate the real number of biases. In the unlikely case that no clue about a good choice of M_{bias} is available, a sensible approach is to run a number of optimisations with different values of M_{bias}. For every value of M_{bias} an average value can be fixed, which represents the expected magnitude of the optimised objective function when no bias has gone undetected. Comparison of the results allows to guess the actual number of biases.

Fig. 4.15 shows the typical convergence of the population of strings to a single fault class. Further generations are necessary to improve the result's accuracy. It is interesting to notice that:

- concentration on the faulty components is easily achieved after few iterations
- the curves are quite smooth and become monotone early in the run.

The fact that concentration is usually achieved early in the run is consistent with Goldberg's finding (Goldberg, 1990). As pointed out in section 4.8.2.1.3, selection strongly dominates early GA performance and restricts the subsequent search to intervals with above-average function value dimension by dimension. This behaviour is probably made more pronounced by the non-linear mapping from objective to fitness function.

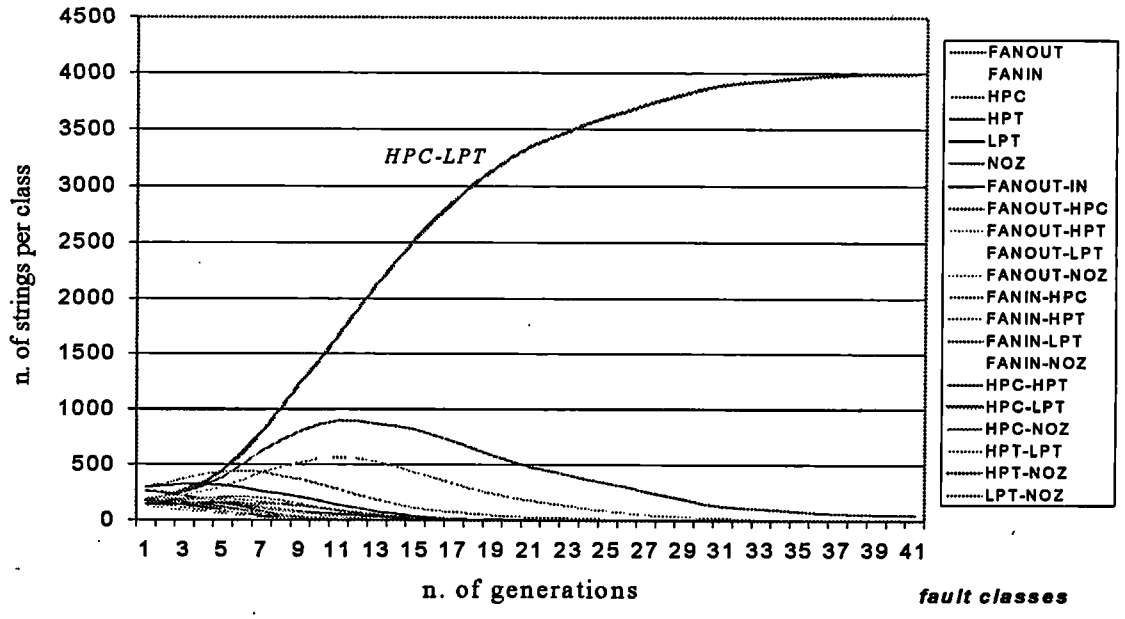


Fig. 4.15: convergence curves for the fault classes

4.10.1 Effect of modelling errors and discrepancies from the noise statistical model

The results presented so far have been obtained with function (4.25)-(4.26) applied to gaussian noised measurements, given the maximum number of biases M_{bias}.
As stated in section 4.7, whenever the gaussian pdf (4.5) can be considered an accurate model of the noise, the following function should be used:

$$J_{klmn}(\mathbf{x},\mathbf{w}) = \sum_{\substack{j=1 \\ j\neq k,l,m,n}}^{M} \frac{[z_j - h_j(\mathbf{x},\mathbf{w})]^2}{(z_{odj}(\mathbf{w})\cdot\sigma_j)^2} \tag{4.59}$$

$$J(\mathbf{x},\mathbf{w}) = \min_{k,l,m,n} J_{klmn}(\mathbf{x},\mathbf{w}) \tag{4.60}$$

where $M_{bias} = 4$.
Function (4.59)-(4.60) is the analogous of (4.25)-(4.26) where squares have been substituted by absolute values.
As a matter of fact, though, the actual measurement noise pdf may not be perfectly gaussian shaped, for a number of reasons:

- in practice the occurrence of readings out of the three standard deviation range is much more common than what is supposed by the gaussian model
- modelling errors are inevitably present, especially in the simulation of gas turbine performance (see sections 1.1 and 4.4).

For these reasons, other objective functions, such as (4.25)-(4.26), are more suitable, as they provide a robust estimation (Huber, 1991; Launer and Wilkinson, 1979; Press et al., 1992). In the sequel a brief introduction to the robustness issue is given.

Statistical inferences are based only in part upon the observations. An equally important base is formed by prior assumptions about the underlying situation. These assumptions are not supposed to be exactly true – they are mathematically convenient rationalisations of an often fuzzy knowledge. The use of these assumptions is usually justified by appealing to a vague continuity or stability principle, according to which a minor error in the mathematical model should cause only a small error in the final conclusions. Unfortunaltely, this does not always hold. Therefore it is important to search for robust statistical techniques, where the term robust means ***insensitive to small deviations from the assumptions***. From a qualitative point of view a robust statistical method should be characterised by the following features:

1) it should have a reasonably good (optimal or near optimal) efficiency at the assumed model
2) it should be robust in the sense that small deviations from the model assumptions should impair the performance only slightly
3) somewhat larger deviations from the model should not cause a catastrophe.

In the problem at hand, the main aim is *distributional robustness*: although the shape of the true underlying distribution deviates slightly from the assumed model (which is

gaussian), the statistical analysis results should not be dramatically affected by such discrepancy.

It is common to simulate the discrepancy from the gaussian noise model by superimposition (with probability ε) to the gaussian pdf of a similar pdf with a standard deviation that is three times larger. The noise cumulative distribution function is then assumed to be made of two terms:

$$F(z;\sigma) = (1-\varepsilon)F_G(z;\sigma) + \varepsilon F_G(z;3\sigma) \tag{4.61}$$

where:

- z is a measurement
- σ is its standard deviation
- ε is a small number (~ 0.1)

F_G is the cumulative gaussian distribution function.

A number of samples are generated according to the distribution function (4.61) and the performance of the two following measures of scatter is compared:

$$d_n = \frac{1}{n}\sum_{j=1}^{n}\left|z_j - \bar{z}\right| \tag{4.62}$$

$$s_n = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(z_j - \bar{z})^2} \tag{4.63}$$

where:

- d_n is the mean absolute deviation
- s_n is the mean square deviation
- n is the number of samples
- $\bar{z}$ is the mean value used in (4.61)
- z_j is the j-th realisation.

The performance of these two different measures of scatter can be properly compared by means of the asymptotic relative efficiency of d_n relative to s_n, defined as follows:

$$ARE = \lim_{n\to\infty} \frac{\mathrm{var}(s_n)/(Es_n)^2}{\mathrm{var}(d_n)/(Ed_n)^2} \tag{4.64}$$

which is simply the ratio of the variances adimensionalised by the squares of the mean values. The larger is its value, the more efficient is the mean absolute deviation relative to the mean square deviation. ARE is actually a function of ε ($ARE = ARE(\varepsilon)$) and can easily be computed. Table 4.3 displays the results according to Huber (1991).

ε	$ARE(\varepsilon)$
0	0.876
0.001	0.948
0.002	1.016
0.005	1.198
0.01	1.439
0.02	1.752
0.05	2.035
0.1	1.903
0.15	1.689
0.25	1.371
0.5	1.017
1.0	0.866

Table 4.3: asymptotic efficiency of mean absolute relative to mean square deviation

A simple glance at table 4.3 suggests that just 2 bad observations in 1000 warrant the mean absolute value better performance. In engineering and physics, typical good data samples appear to be well modelled by an error law like (4.61) with ε in the range between 0.01 and 0.1. Therefore, d_n, even though less easily manageable, should be preferred to s_n, in that more robust.

It can be shown that if the quantity $\hat{z}$ is estimated by minimising the functions:

$$d_n(\hat{z}) = \frac{1}{n}\sum_{j=1}^{n}\left|z_j - \hat{z}\right| \tag{4.65}$$

$$s_n(\hat{z}) = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(z_j - \hat{z})^2} \tag{4.66}$$

the following results are respectively obtained:

$$\hat{z} = z_{med} \tag{4.67}$$

$$\hat{z} = z_{mean} \tag{4.68}$$

where:

- z_{med} is the median
- z_{mean} is the mean.

Thus, an estimation based on the minimisation of the mean absolute deviation (and hence on the median) can be regarded as more robust.

In the case of the diagnostic problem at hand, minimising the objective function (4.59)-(4.60) means searching for the mean value, while if function (4.25)-(4.26) is used the

estimate should be close to the median. A set of test cases have been run with both (4.25)-(4.26) and (4.59)-(4.60) to compare the diagnostic accuracy when the measurement noise is not perfectly gaussian. 50 mixed test cases have been run to compare the two objective functions when $\varepsilon = 0.3$ in the model (4.61) and hence the discrepancy is assumed to be significant.
Using the objective function (4.25)-(4.26) is like assuming the measurement noise to be defined by a double exponential joint pdf:

$$p(\mathbf{v}) = \frac{1}{\left(\sqrt{2}\right)^M} \prod_{j=1}^{M} \left(\frac{1}{\sigma_j}\right) e^{-\sqrt{2}\sum_{j=1}^{M} \frac{|v_j|}{\sigma_j}} \tag{4.69}$$

instead of (4.5). This pdf has much wider tails than the gaussian pdf for the same value of the standard deviation*. Fig. 4.16 plots a double exponential and two gaussian pdfs.

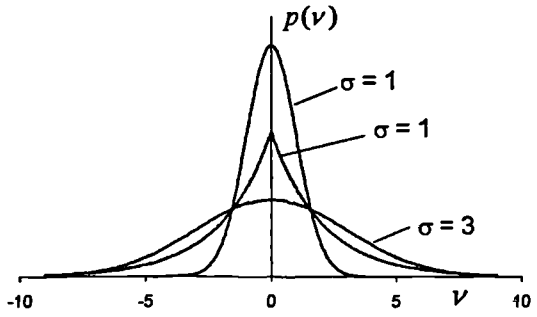


Fig. 4.16: probability density functions

A look at fig. 4.17 confirms that the double exponential pdf is suitable for approximating the discrepancy typically modelled by (4.61).
It is worth highlighting that the minimisation of functions like (4.25)-(4.26) and (4.59)-(4.60) actually leads to *local M-estimates* (Huber, 1991; Launer and Wilkinson, 1979; Press et al., 1992). In this perspective, the choice of different objective functions means that different pdfs are assumed. Pdfs different from the two which have been used in this study could be chosen (e.g. the Lorentzian, the Andrew's sine, the Tukey's biweight, etc). Their distinctive feature is usually represented by wider tails than the classic gaussian pdf.

Table 4.4 summarises the results of the comparison study made with the EJ200 model.

* Utilisation of (4.25)-(4.26) is statistically sound if σ_j is the mean absolute deviation and not the standard deviation. However, measurement non-repeatability of gas turbine sensors is usually expressed by standard deviations. Moreover if the noise is gaussian, the absolute mean deviation converges to $\sqrt{2/\pi}\sigma$ if the number of observations made to estimate the noise scatter is large

	func. (4.59)-(4.60)	func. (4.25)-(4.26)
RMS	0.487	0.198
e_1 (g/s)	3.30	1.80
e_2 (kN/m^2)	0.51	0.27
e_3 (K)	0.83	0.58
s.c.c. (%)	63.6	72.7
s.s.c. (%)	86.4	86.4

Table 4.4: comparison of results with different objective functions

As expected, the objective function (4.25)-(4.26) provides remarkably more accurate results due to its robustness to small deviations from the assumed model. The estimation errors are smaller and the capability to isolate faulty components better. However, the two functions show the same capability to isolate faulty sensors.
As far as the optimisation process is concerned, use of the smoother function (4.59)-(4.60) allows a faster convergence, whereas function (4.25)-(4.26) is harder to minimise. Actually the main reason why in the past the mean absolute deviation has often been discarded in favour of the mean square deviation is just the difficulty of dealing with such a highly non-smooth functions. Use of the EP-based optimiser allows overcoming the problem.

Two remarks still have to be made:

- strictly speaking, the statements on statistical robustness just made apply when the number of samples to rely on is large. Usually, the numbers of measurements available for gas turbine diagnostics are few. Thus, these statistical considerations have to be accepted carefully. However, the experimental results seem to confirm the validity of the approach
- so far, only discrepancies from the statistical model have been discussed. However, another relevant source of uncertainty is model errors. Using a robust approach can help reduce the effects of these unavoidable errors on the overall estimation accuracy.

4.11 The RB199 engine and instrumentation

Another engine used for testing the diagnostics is the three-spool low by-pass ratio military turbofan engine RB199 (Zedda and Singh, 1999c). It is the powerplant of the PANAVIA Tornado multi-role combat aircraft. It has been chosen due to the following reasons:

- it is representative of the complex three-spool engine configuration (i.e. the Trent family)
- real data can be collected by the sponsoring company in order to validate the method properly.

The RB199 is a low by-pass ratio mixed flow reheated turbofan. Three spools, it is 74 kN thrust class. Table 4.5 displays some cycle parameters.

by pass ratio	1.02
overall pressure ratio	23
combat (max reheat) thrust	74.1 kN
max dry thrust	42.5 kN

Table 4.5: RB199 cycle parameters

Fan, intermediate and high-pressure compressors are three-stage. The combustion system is annular and the three turbines are single stage.
The RB199 engine's health is modelled by 8 components and like all low by pass ratio engines a single total mass flow graph is used. The components are fan outer, fan inner, IP compressor, HP compressor, HP turbine, IP turbine, LP turbine and propelling nozzle. Fig. 4.17 shows the model's schematic. In the picture, the dashed line is used for components whose performance is defined by constant parameters that are not to be estimated.

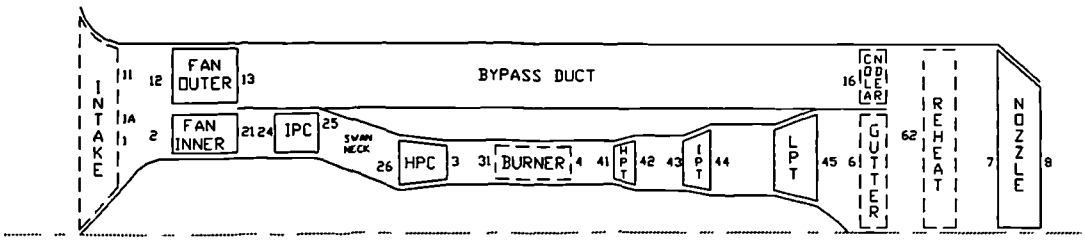


Fig. 4.17: schematic of the aero-thermodynamic model of the RB199

Thus, the following performance parameters (14) express the health status of the engine:

- fan overall flow function (Γ_{FAN})
- fan outer efficiency (η_{FANOUT})
- fan inner efficiency (η_{FANIN})
- IP compressor flow function and efficiency (Γ_{IPC}, η_{IPC})
- HP compressor flow function and efficiency (Γ_{HPC}, η_{HPC})
- HP turbine flow function and efficiency (Γ_{HPT}, η_{HPT})
- HP turbine flow function and efficiency (Γ_{IPT}, η_{IPT})
- LP turbine flow function and efficiency (Γ_{LPT}, η_{LPT})
- propelling nozzle discharge coefficient (C_D).

16 measurements used for the analysis are those available in the test facility (monitoring measurements):

- engine inlet airflow (W_{1A})
- fan outer exit total pressure and temperature (P_{13}, T_{13})
- fan inner exit total pressure and temperature (P_{21}, T_{21})
- core inlet mass flow (W_{21})
- IP compressor exit total pressure and temperature (P_{25}, T_{25})
- HP compressor exit total pressure and temperature (P_3, T_3)

- LP turbine exit total pressure and temperature (P_{45} , T_{45})
- thrust (F)
- spool speeds (N_H, N_I , N_L).

This set of measurements has been chosen according to a work carried out by Willan (1990) with real test data.
3 parameters are used to set the operating point of the engine:

- main burner fuel flow (W_{FE})
- ambient total pressure and temperature (P_0, T_0).

The model used for simulation has been provided by Rolls-Royce. It is a non-linear steady state performance simulation program named RRAP as well as the one used for the EJ200 simulations.

4.12 Testing with the RB199

The diagnostic method has already been applied to a two-spool military turbofan engine, the EJ200 (see section 4.10 and Zedda and Singh, 1999a). The present work deals with the application of fault diagnosis to a more complex engine, the three-spool military turbofan RB199. 1 or 2 engine components are supposed to be faulty ($N_{perf} = 1 \div 4$) in presence of 2 or 5 faulty sensors ($M_{bias} = 2$ or $M_{bias} = 5$). The GA population is made of 36 fault classes (8 classes for faults in a single component and 28 classes for faults in two components). The 3 environment and power setting parameters are both noisy and biased. Such a large number of biased measurements (up to 8 biases in 19 measurements) is chosen in order to assess the system's capability to deal with sensor faults. In a real development test bed, this kind of instrumentation set could be affected by just two or at most three biases simultaneously.
The magnitude of the biases has been set to 1% for all measurements, apart from those whose assumed measurement *non-repeatability range, equal to* $3 * \sigma$, *is comparable to* 1% variation. For these, a level of 2% has been chosen. The biases affecting the environment and power setting measurements have been set to the large value of $50 * \sigma$ as accommodation is automatically carried out.
The maximum level of deterioration has been set to 3%. Larger deterioration levels can be dealt with simply by expanding the range of variation of the performance parameters in the GA optimiser.
High power, dry operating points are analysed.
72 test cases have been run to assess the system's accuracy. For every class a certain fault has been analysed with 2 and 5 measurement biases. A difference with respect to the EJ200 simulations is that here measurements that would definitely be useful to diagnose the fault have been biased on purpose. Spool speed measurements were noisy but bias-free.
The discrepancy from the gaussian noise model (see section 4.10.1) is again simulated by superimposition (with probability ε) *for every test case* to the correct gaussian pdf of a similar pdf with a standard deviation which is three times larger. In the present work, ε has been set to 0.3.

To be on the safe side, the population is made of 1800 strings. However a population of 900 strings is usually sufficient to avoid local minima and reach good accuracy. 100 iterations are definitely sufficient to perform the optimisation.

Fig. 4.18 shows typical results for two faulty components. Concentration on the faulty components is achieved and faults are accurately quantified. The biases in the environment and power setting parameters are accommodated.

parameters	actual (%)	predicted (%)
$\Delta\eta_{FANOUT}$	0.	0.
$\Delta\Gamma_{FAN}$	2.	2.00
$\Delta\eta_{FANIN}$	-1.	-0.99
$\Delta\Gamma_{IPC}$	0.	0.
$\Delta\eta_{IPC}$	0.	0.
$\Delta\Gamma_{HPC}$	0.	0.
$\Delta\eta_{HPC}$	0.	0.
$\Delta\Gamma_{HPT}$	0.	0.
$\Delta\eta_{HPT}$	0.	0.
$\Delta\Gamma_{IPT}$	3.	2.99
$\Delta\eta_{IPT}$	-3.	-2.94
$\Delta\Gamma_{LPT}$	0.	0.
$\Delta\eta_{LPT}$	0.	0.
C_D	0.	0.
		RMS = 0.02
W_{FE}	479.7	478.4
P_{AMB}	96.539	96.794
T_{AMB}	285.8	286.2

Fig. 4.18: typical results for a 2 faulty component test case

Fig. 4.19 shows the isolation of 5 measurement biases. The numbers shown are the values of the terms corresponding to the various measurements to be added up in the objective function. The measurements marked with "*" have been generated by assuming a probability density function with a three times larger standard deviation.

Table 4.6 summarises the results for the 72 test cases run. The same remarks made for the EJ200 apply here.
In general, the larger the relative redundancy (and then $(M-M_{bias})/(N_{perf}+P)$) the higher the accuracy. In this respect, it is worthwhile to point out that the results just presented are slightly better than the ones obtained with a two spool turbofan engine EJ200 (see tables 4.2 and 4.4), especially for the test cases with a large number of measurement biases. This is due to a larger value of the relative redundancy ratio quoted above.

	actual	predicted
W_{1A}	*1.99	2.16
P_{13}	1.29	1.26
T_{13}	2.07	1.11
P_{24}	10.04	9.62
T_{24}	10.86	9.8
W_{24}	0.26	0
P_{26}	1.08	1.33
T_{26}	0.19	0.77
P_3	0.71	0.67
T_3	24.48	24
P_{45}	13.24	13.78
T_{45}	32.45	33.87
F	*3.24	0.11
N_H	0.43	0.45
N_I	0.78	0.06
N_L	2.12	0.25

Fig. 4.19: bias isolation

	2 biases	5 biases
1 faulty component		
RMS	0.07	0.08
e_1 (g/s)	0.6	0.7
e_2 (kN/m^2)	0.09	0.17
e_3 (K)	0.2	0.3
s.c.c. (%)	88	88
s.s.c. (%)	94	90
2 faulty components		
RMS	0.09	0.16
e_1 (g/s)	0.8	0.8
e_2 (kN/m^2)	0.21	0.36
e_3 (K)	0.5	0.8
s.c.c. (%)	95	93
s.s.c. (%)	94	98

Table 4.6: test case results

The relatively low values of s.c.c. for the one faulty component cases are simply due to the limited number of test cases considered: in one test case (with both 2 and 5 biases) the faulty component was isolated but a small fault was found even in another component. This "smearing" effect, though, is seldom present and affects the RMS to a very limited extent.

Fig. 4.20 and 4.21 show the comparison between the minimum objective function value and the RMS vs. the number of generations. The similarity of the two curves, especially in a late phase of the convergence, confirms that the objective function has been chosen adequately.

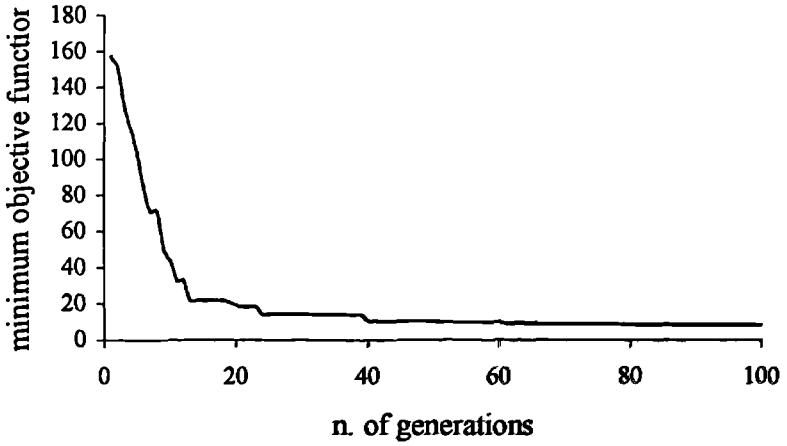


Fig. 4.20: minimum objective function value vs. number of generations

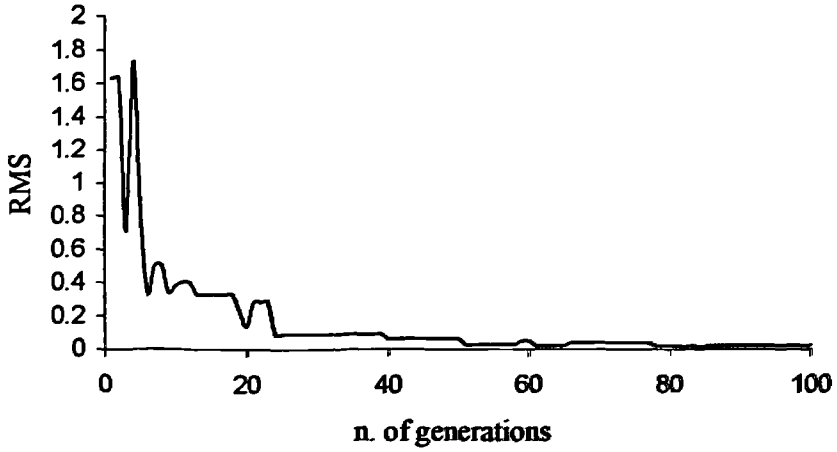


Fig. 4.21: RMS vs. number of generations

Fig. 4.22 plots the number of strings vs. the number of generations for every fault class in the case considered in fig. 4.19. Here the legend on fault classes is not reported for clarity of representation. 900 strings were sufficient to reach the minimum. As shown in the figure, a competition occurred in the first phase of the convergence among the fault classes. After about 30 iterations the winning class is clearly the one representing faults in fan inner and IP turbine.

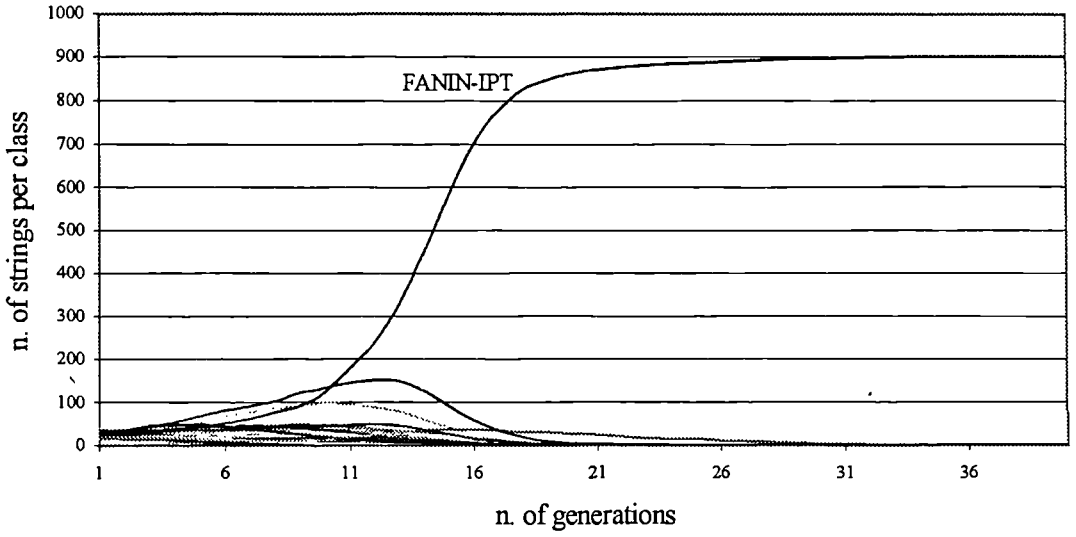


Fig. 4.22: convergence curves for the fault classes

4.13 Multiple Operating Point Analysis of poorly instrumented engines

The optimisation-based diagnostics developed so far suits well the performance analysis of comprehensively instrumented engines. Specifically, the method has been devised to deal with development engines which are usually well equipped with measurement instrumentation. On the contrary, pass-off tests rely on a much poorer instrumentation set.

A simple modification of the optimisation-based fault diagnosis technique just introduced allows to perform Multiple Operating Point Analysis (MOPA). MOPA's key idea is to analyse a number of operating points simultaneously in order to extract the maximum amount of information from the available instrumentation set. Some remarks already introduced in section 2.3 are here repeated for clarity. Better accuracy is claimed when the operating conditions are far from each other (Stamatis and Papailiou, 1988). The method's accuracy has actually been always tested with simulated data. When deteriorated engine performance is simulated, as no exact information about the actual map's modification is usually available, the fault's effect is obtained by a simple shift of the map itself. If this simplification is accepted, utilisation of operating points located far from one another on the map should enable maximum exploitation of the model non-linearity and then of the available instrumentation set. The technique's underlying assumption is obviously that the value of the performance parameter variation is not modified by changing operating condition. On the other hand Doel (1994b) noted that a different working point means different aerodynamic conditions and in this sense efficiencies and flow capacities can significantly change with the operating condition. The deviation though can be supposed small provided the power settings are close to each other. That is why when analysing real data the operating conditions should be close to one another.

Adaptation of the optimisation technique to MOPA is obtained by summing up the objective functions corresponding to the various operating points.
A vector $\mathbf{W}$ is given by juxtaposition of L operating points:

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}^{(1)} \\ \mathbf{w}^{(2)} \\ \cdot \\ \cdot \\ \mathbf{w}^{(L)} \end{bmatrix} \tag{4.70}$$

where $\mathbf{w}^{(j)}$ is the vector of environment and power setting parameters for the j-th operating point. If no monitoring measurement biases were present, the objective function to minimise would become:

$$J(\mathbf{x},\mathbf{W}) = \sum_{k=1}^{L}\sum_{j=1}^{M} \frac{\left|z_j^{(k)} - h_j(\mathbf{x},\mathbf{w}^{(k)})\right|}{z_{odj}(\mathbf{w}^{(k)})\cdot\sigma_j} \tag{4.71}^*$$

If a number of biases are supposed to affect the monitoring measurement set (e.g. $M_{bias} = 4$), during the optimisation procedure evaluation of the objective function is made according to the following algorithm:

$$J_{lmnp}(\mathbf{x},\mathbf{W}) = \sum_{k=1}^{L} \sum_{\substack{j=1 \\ j\neq l,m,n,p}}^{M} \frac{\left|z_j^{(k)} - h_j(\mathbf{x},\mathbf{w}^{(k)})\right|}{z_{odj}(\mathbf{w}^{(k)})\cdot\sigma_j} \tag{4.72}$$

$$J(\mathbf{x},\mathbf{W}) = \min_{lmnp} J_{lmnp}(\mathbf{x},\mathbf{W}) \tag{4.73}$$

In the single operating point analysis, for the technique to apply the inequality (4.24), reported here for convenience, has to be satisfied:

$$M - M_{bias} > N_{perf} + P \tag{4.74}$$

In case of MOPA, relative redundancy is obtained when:

---

* For temperature measurements the noise standard deviation is a function of the temperature itself. Therefore, a correct treatment of noise would call for different values of the standard deviation for the same temperature at different operating points. However, as the operating points considered are usually quite close to one another, this effect can be safely neglected and the same σ_j is used for all operating conditions.

$$L \cdot (M - M_{bias}) > N_{perf} + L \cdot P \tag{4.75}$$

where P is the number of parameters used to define the operating point of the engine (i.e. $P = 3$). If a *relative redundancy index* is defined as follows:

$$R = \frac{L \cdot (M - M_{bias})}{N_{perf} + L \cdot P} \tag{4.76}$$

it is possible to study the effect of using increasing numbers of operating points for the same diagnostic case. It is worth reminding that the assumption under which both (4.74) and (4.75) apply is that all fault-affected performance parameters and environment and power setting parameters depend on all bias free monitoring measurements. As this assumption is clearly a simplification and moreover the stochastic nature of the problem and the different levels of noise affecting the various measurements must be taken into account, the following considerations have to be regarded as approximate.
The following case, which can be regarded as typical, is considered: 4 performance parameters are fault-affected ($N_{perf} = 4$), the operating point is defined by 3 parameters ($P = 3$), 10 measurements are used for monitoring ($M = 10$) and two of them are biased ($M_{bias} = 2$). If R is plotted as a function of the number L of operating points, fig. 4.23 is obtained.

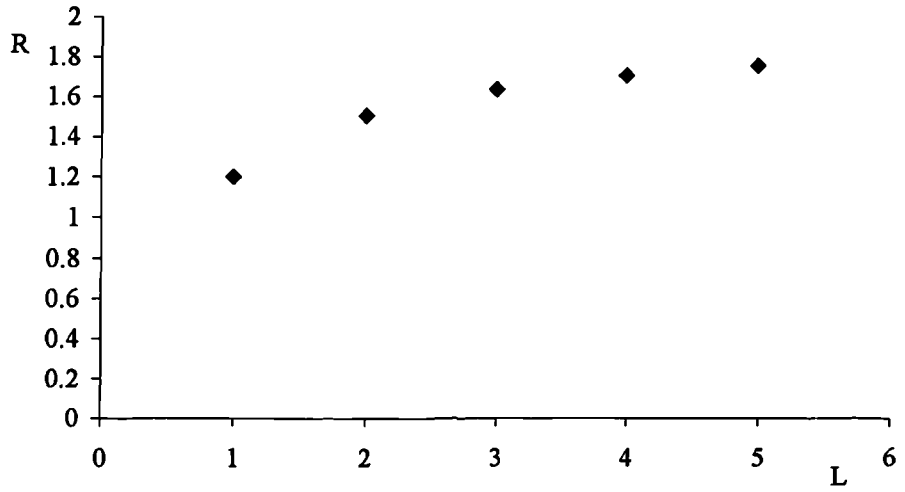


Fig. 4.23: relative redundancy index vs. number of operating points

As expected, an increase in the number of operating points increases the relative redundancy index and hence the method's reliability.
It is interesting to study the effect on relative redundancy of a larger number of operating points. The corresponding plot is given in fig. 4.24.

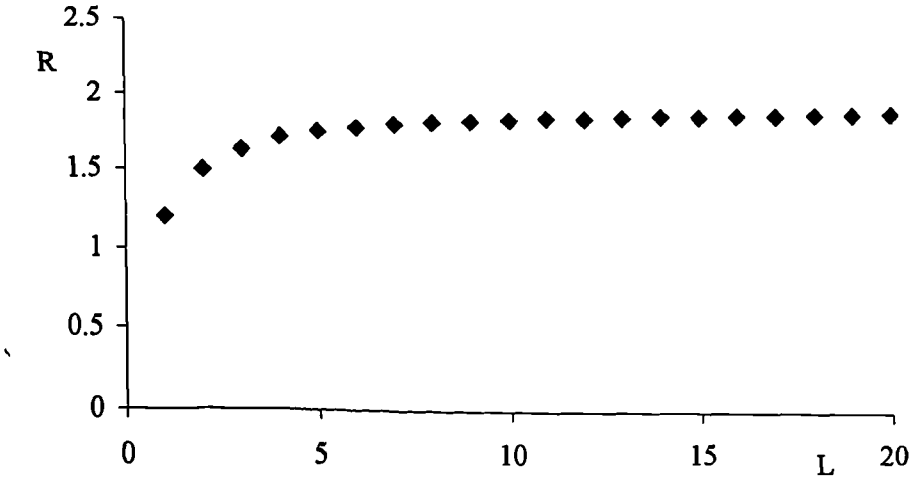


Fig. 4.24: effect of using a large number of operating points

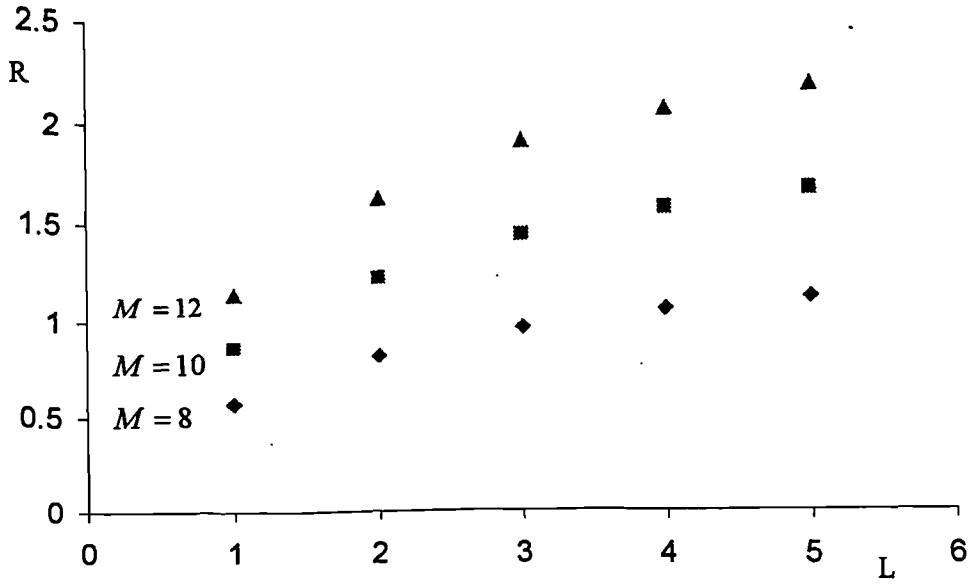


Fig. 4.25: effect of the number of measurements on relative redundancy

A substantial increase of the number of operating points carries the following drawbacks:

- the rate of increase of the relative redundancy index reduces by increasing the number of operating points. Eventually, fig. 4.25 shows that the relative redundancy index tends to an asymptote
- the computational resources necessary to evaluate the objective function increase with the number of operating points. When $L = 20$ an evaluation may require a time 20 times longer than with a single operating point. Thus the optimisation problem may become unmanageable

- a very large number of operating points is likely to cover a rather wide area in the component maps. Therefore, the assumption of constant performance parameter variations is less acceptable
- the objective function becomes harder to optimise as the number of points increases.

In conclusion, an increase in the number of operating points is advisable up to a certain extent. Fig. 4.23 suggests that the gains achievable for low numbers are remarkable.

Fig. 4.25 shows that an increase in the number L of operating points allows to increase the relative redundancy index. For instance, the use of 2 operating points ($L=2$) for an instrumentation set made of 10 measurements ($M=10$) produces a relative redundancy index slightly larger than the one corresponding to single operating point analysis ($L=1$) with 12 measurements ($M=12$).
Fig. 4.26 shows the effect of the number of biases on the relative redundancy index.

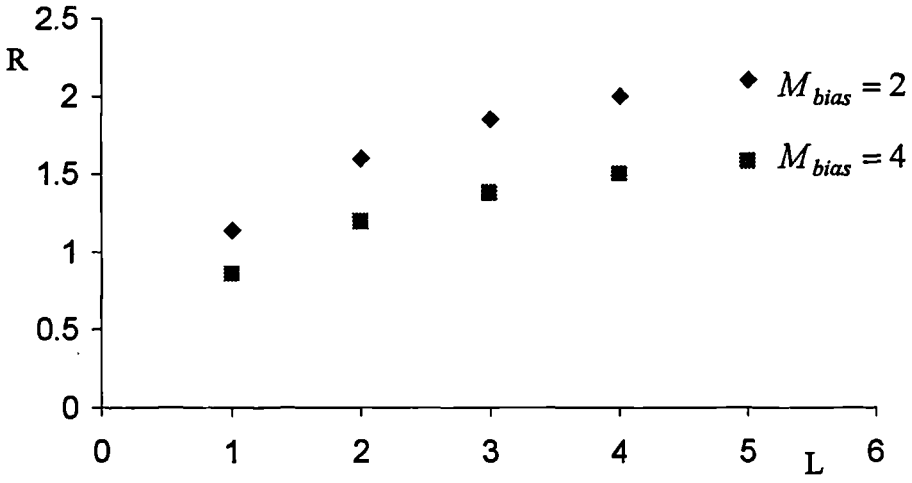


Fig. 4.26: effect of the number of biases on relative redundancy index

As expected, a decrease in the number of biases increases the relative redundancy index. A similar effect is given by a decrease in the number of fault-affected performance parameters.

The theoretical considerations and practical results on relative redundancy (see section 4.12 and Zedda and Singh, 1999c) show that additional information can be obtained by analysing more operating points simultaneously. However, optimisation of the objective function defined by (4.72)-(4.73) is much more complicated. This is due to the following effects:

- the number of variable parameters is larger and thus the problem's dimensionality increases
- the increase in the number of environment and power setting parameters to be estimated makes the shape of the objective function more complex. As mentioned in section 4.8, environment and power setting parameters affect most of the terms in the summation both in the denominator and numerator
- evaluation of the objective function is heavier from a computational point of view.

In order to understand the typical problems encountered when trying to minimise an objective function, it can be useful to visualise it, although the optimisation problem is highly multi-dimensional and only three-dimensional visualisations are possible.
Fig. 4.27 and 4.28 show a plot of the objective function for varying HP compressor efficiency and flow function. The EJ200 model has been used and the HP compressor and LP turbine have been considered faulty. 2 operating points have been used and 2 measurements were faulty. Parameters which are not shown were constant and equal to the corresponding actual value.
Fig. 4.29 shows the plot of a function which has the minimum value when the objective function's evaluation involves elimination of the actually biased measurements. Moreover, different combinations of discarded measurements correspond to different levels. The tailored function has been called *measurement validation function* (m.v.f.).
Fig. 4.30 and 4.31 show a plot of the objective function when fuel flow and ambient pressure are varying.
Fig. 4.32 and 4.33 show two different views of a plot of the measurement validation function with varying environment and power setting parameters.

Notwithstanding the limitation of the 3-D plotting, the following hints can be extracted:

- the objective function is rather smooth when considered as a function of the performance parameters only
- when the environment and power setting parameters are varying as well, the objective function is more rugged and hence more difficult to optimise
- a glance at the measurement validation functions shows that biased measurement isolation is much more complicated when environment and power setting parameters are allowed to vary. Jumps in the measurement validation function lead to discontinuities in the objective function's first derivatives. The more complicated is the measurement validation function (in terms of hills and canyons), the less smooth is the objective function.

In conclusion, optimising the objective function for multiple operating point analysis seems more complicated than for single operating point analysis due to the larger number of unknown environment and power setting parameters and the related topological problems.

A sensible way to tackle the problem is to take into account the actual probability of occurrence of biases in the environment and power setting parameters. As a matter of fact, whereas fuel flow measurement biases are quite common, biases in ambient pressure and temperature are very rare and can easily be spotted. Hence, the range of variation of ambient pressure and temperature in the EP can be significantly reduced to only account for noise. Assuming ambient and pressure measurements to be noisy but not biased is sensible from a diagnostic point of view and very useful from an analytical point of view. Direct consequence of this is that the MOPA objective function becomes smoother and hence easier to optimise.

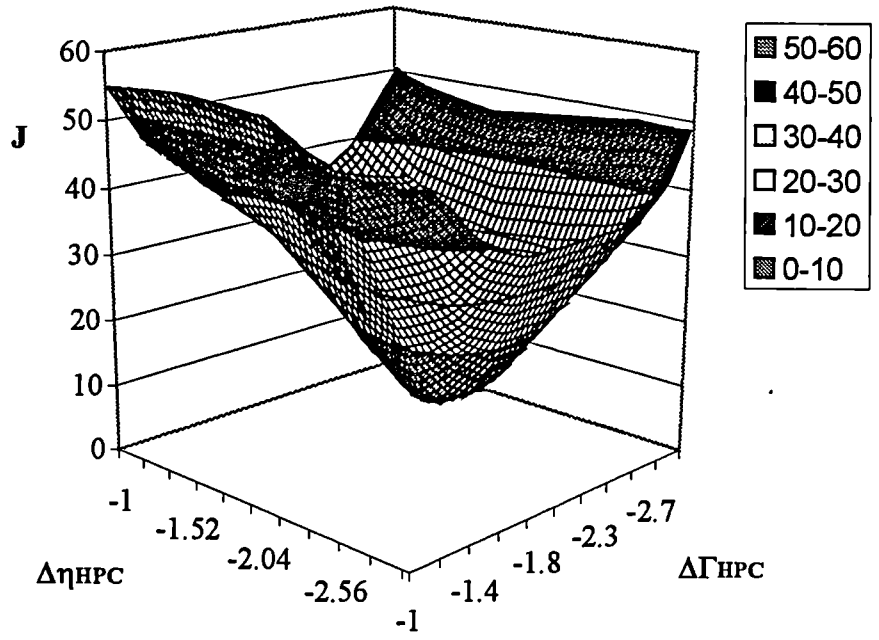


Fig. 4.27: objective function vs. HP compressor performance parameters (view 1)

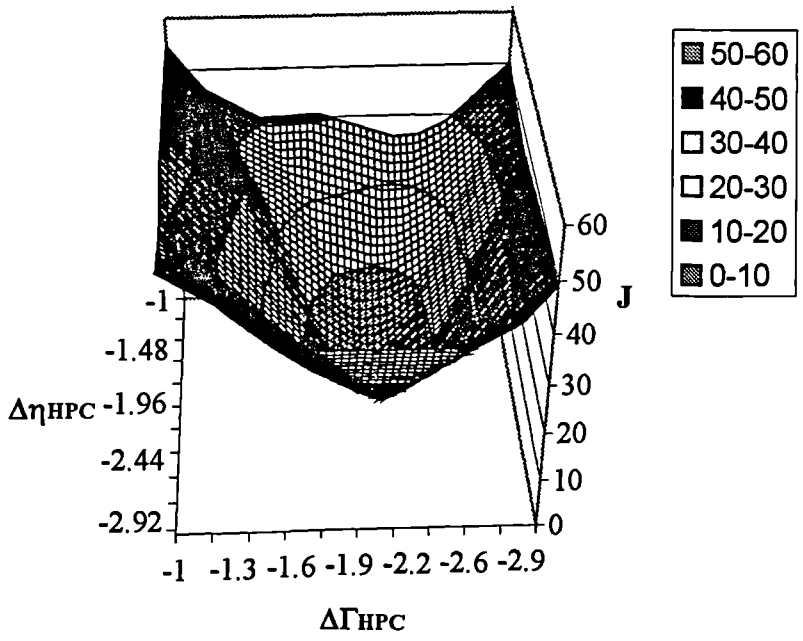


Fig. 4.28: objective function vs. HP compressor performance parameters (view 2)

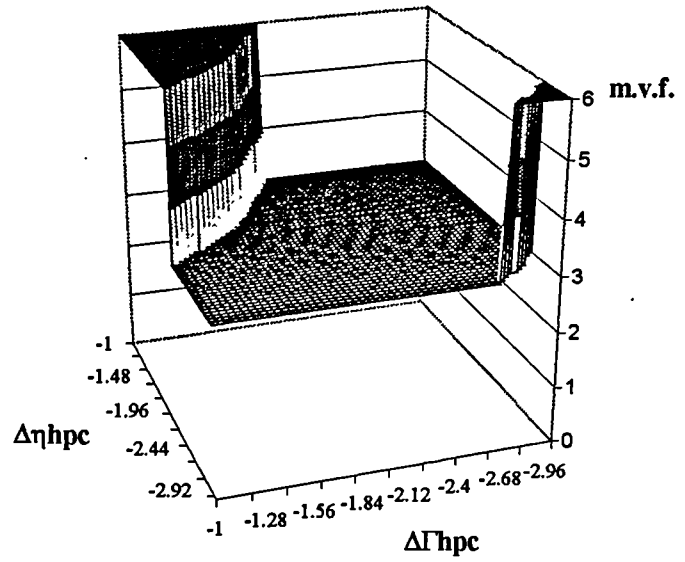


Fig. 4.29: measurement validation function vs. HP compressor performance parameters

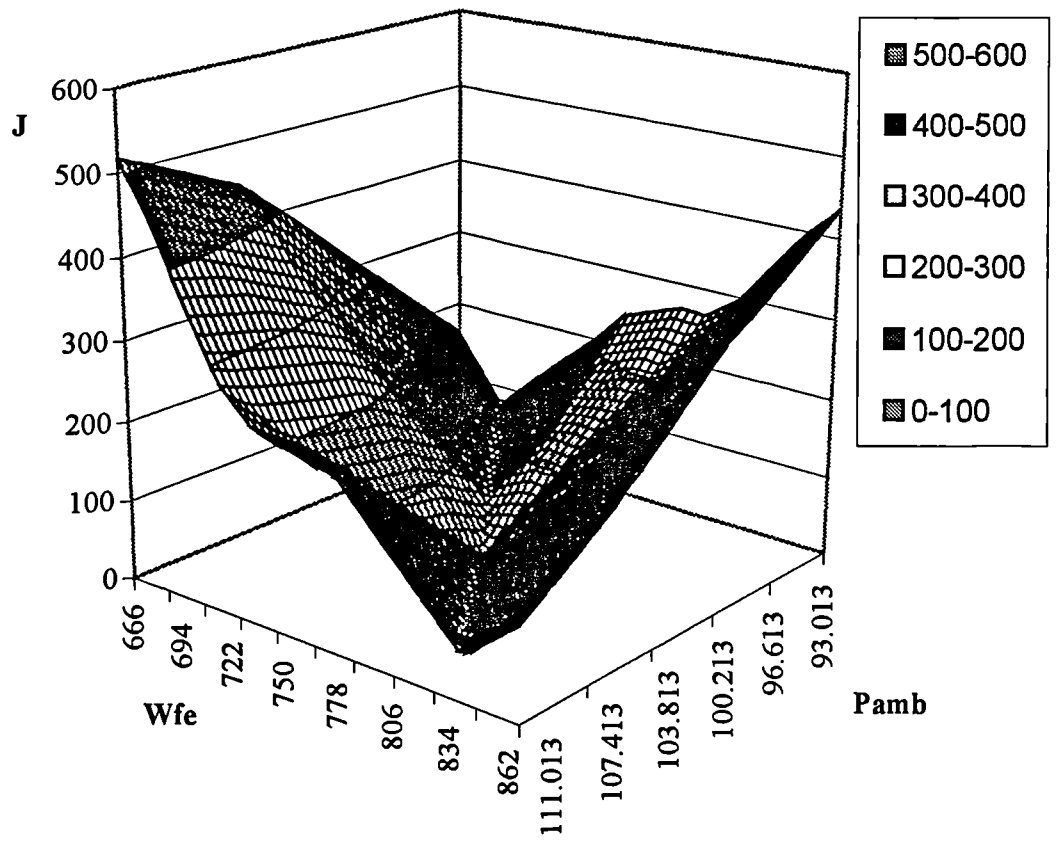


Fig. 4.30: objective function vs. environment and power setting parameters(view 1)

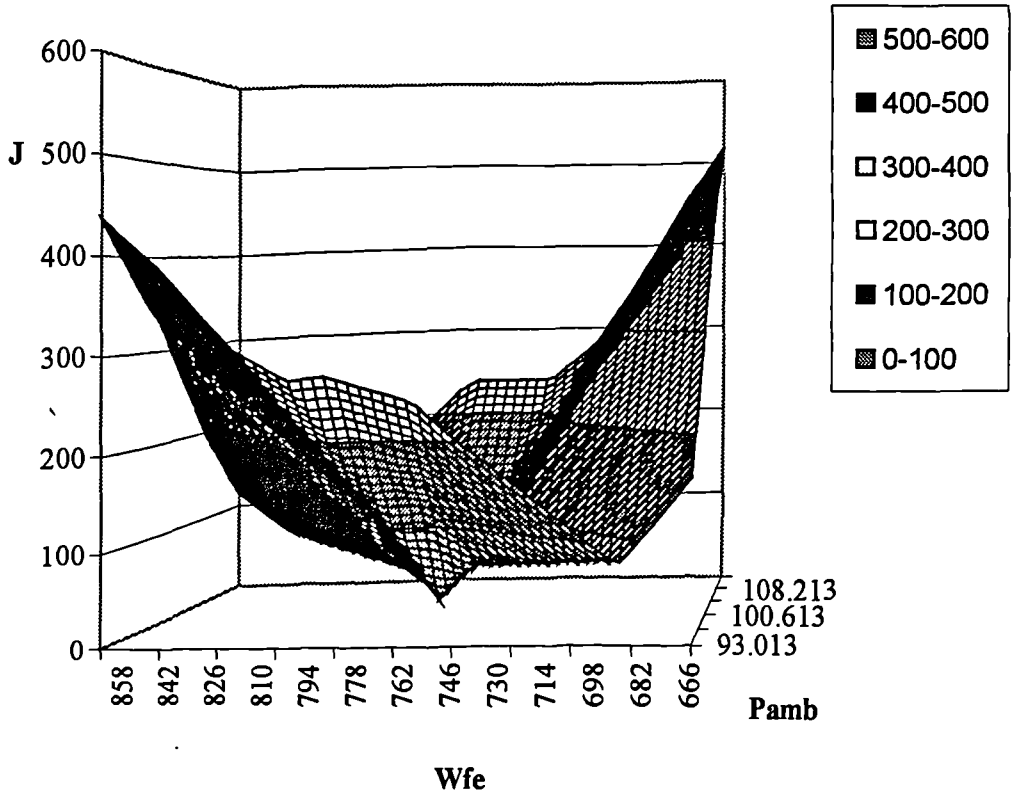


Fig. 4.31: objective function vs. environment and power setting parameters(view 2)

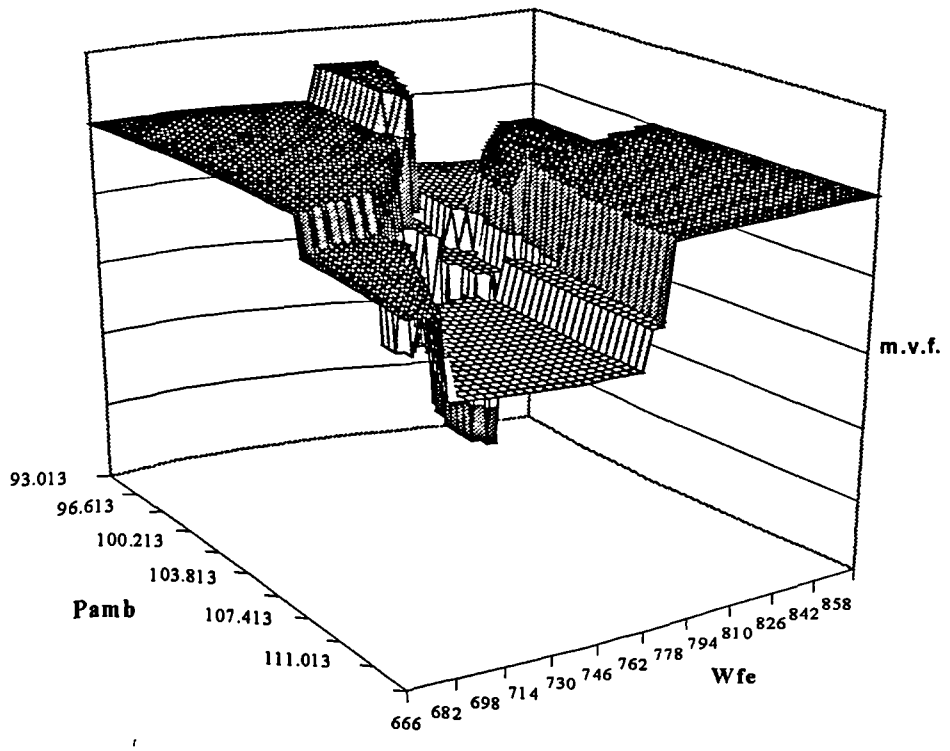


Fig. 4.32: measurement validation function vs. environment and power setting parameters (view 1)

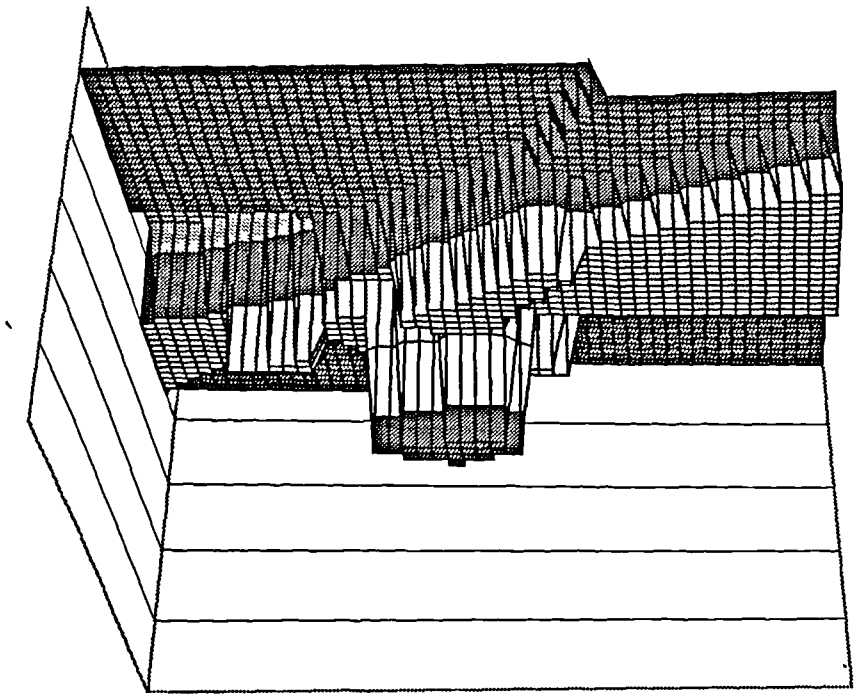


Fig. 4.33: measurement validation function vs. environment and power setting parameters (view 2)

Even so, a straightforward application of the GA-based optimiser described in sections 4.9 actually revealed that problems of convergence are not uncommon. Most of the times, the optimiser got stuck at local minima corresponding to an incorrect choice of faulty engine components. Even when the faulty components were isolated, the optimiser showed not to be able to "run down the hill" and reach the global minimum. Therefore, the simulation findings basically confirmed the theoretical remarks made above.
Two different avenues can be pursued to improve the capability of the GA-based optimiser:

- use the GA along with a suitable hill climbing technique to improve its local search capability
- use multi-objective optimisation.

In the next section, the development of the enhanced GA by means of local search techniques is described.

4.13.1 Enhanced GAs for MOPA

Enhancement of the GA-based optimiser by means of hill climbing methods can be made in two different ways:

- the GA is run and the best string found during the run (or a set of high performance individuals) is used as starting point for the hill climbing procedure
- hill-climbing is used together with the GA during the GA run.

Utilisation of the hill climbing technique after the GA is not suitable for the problem at hand because useful strings can get lost even early in the run if the GA capability to deal with rugged and multi-modal hypersurfaces is inadequate. As the simulations have shown, sometimes the solution proposed by GA at the end of the run corresponds to an incorrect fault class, whilst the right fault class may have died off quite early in the run. In this case, application of a hill climbing technique will not improve the accuracy of the diagnostic answer.
Therefore, the proposed method envisages the concurrent use of GA and proper hill climbing methods.
The way the hill climbing techniques have been used is as follows: every n GA generations, the best string for every fault class is used as starting point for the hill climbing method. The result so achieved substitutes for the initial string. Usually $n=10$. For hill climbing methods that require a set of strings, see section 4.13.1.2 for details.

4.13.1.1 Enhancement by calculus-based methods

Conventional, calculus-based techniques have been tested to ascertain their suitability to help the GA improve its performance. Both the downhill simplex and the Powell's method with discard of the largest descent direction have been used (see section 4.8.1). As expected, better results have been obtained with the simplex method, due to the inherent unsuitability of the Powell's technique to deal with the objective function's non-smoothness. As the hill climbing technique was used for each fault class separately, the problem of handling the constraint on the number of fault-affected performance parameters was automatically overcome. The constraint on the allowed range of variation of the performance parameters was satisfied by means of the penalty term technique as by (4.35). The algorithm was let run until convergence. The following remarks can be made:

- application of the simplex technique does not significantly increase the computational burden, as convergence to some solution is easily reached
- application of the penalty term method for handling the range constraint is relatively successful when the initial point is far from the limits of variation. Whenever the starting point is close to the bound, though, points may be generated that are either out-bound or even unfeasible (e.g. increasing the component efficiency)
- the algorithm seems to be unable to jump from a certain selection of biased measurements to another. If the starting point is such that the selection of biased measurements is the correct one, then the resulting solution proposed by the simplex at the end of its run is substantially better than the initial one. If, on the contrary, the initial set of measurements excluded from the objective function's calculation is wrong, the simplex technique finds it difficult to identify the correct set and then the resulting solution is not significantly better than the initial one.

The outcome of the application of the simplex method is that the technique is rather inefficient due to the objective function's inherent characteristics.

4.13.1.2 Enhancement by Evolution Strategies

Various types of ESs (see section 4.8.2.3) have been applied to enhance the real-coded GA. ESs seem to be suitable to tackle the optimisation problem at hand due to the following reasons:

- constraints: the barrier method is straightforward and suits the ES mechanism of optimisation well
- function's non-smoothness: the technique is robust in this respect. Even highly non-well-behaved functions like (4.72)-(4.73) should be amenable to treatment by ESs
- number of strings: the small population size required by typical ESs suits well the need to perform optimisation with a limited number of strings corresponding to the lower objective function strings for each fault class.

Two-membered, $(\mu+\lambda)$ and (μ,λ)-ESs have been implemented and tested. After the first 10 GA generations, the ES is utilised every 10 generations. A fixed number of ES iterations is performed.
For the two-membered ES, simply the best string for each fault class is used as the first parent. For multi-membered ESs a number μ of parents is given. These strings can be obtained in either of the two following ways:

- the best string for each fault class is selected and is subject to mutation to create a population of μ parents
- the best μ strings for each fault class are selected and used as the initial parents' population.

By testing both methods, the second one turns out to be the best, in that the greater diversity in the initial population is useful in the following search procedure. Therefore all multi-membered ESs described in the sequel use this initial population's selection criterion.

The 1/5 success rule two-membered ES turns out to have a limited capability of optimisation. Constraints can easily be dealt with but the objective function's ruggedness hinders a successful optimisation. This method's performance can be considered similar to the simplex method's one, apart from the issue of constraints. The variances' deterministic adaptation rule is likely to be responsible for the problems encountered in optimisation. Premature convergence is rather frequent and the gains in accuracy of the results are minimal.

Better performance has been achieved by $(\mu+\lambda)$-ESs. The ES parameters were set as follows:

- $\mu=5$
- $\lambda/\mu=5$
- initial σ's are chosen as fractions of the corresponding variation range.

The performance of the $(\mu+\lambda)$-ES has been tested by means of a number of simulations that confirmed the typical problems affecting this optimisation tool. In particular, after a few iterations the ES often gets stuck and successive generations are unable to optimise the function further.

(μ,λ)-ESs have been tested to realise if any improvement was possible. The same method's parameters as used for the $(\mu+\lambda)$-ES were chosen. The (μ,λ)-ES performs much better than the $(\mu+\lambda)$-ES: the phases of recession are short and the optimisation is effective.

Eventually, correlated mutations are tested. This refined ES technique should be well suitable for optimising functions with narrow and twisty valleys not necessarily aligned with the axes.
As far as the objective function at hand is concerned, the following remarks can be made:

- the presence of absolute values in the objective function definitely makes the valleys narrow
- the presence of a large number of varying environment and power setting parameters makes the function more rugged
- even in simple cases such as the one displayed in fig. 4.30 and 4.31, the valley to be reached is not aligned to the axes.

In the light of the points made above, a (μ,λ)-ES with correlated mutations has been implemented and tested according to (4.51)-(4.55). If the number of variable environment and power setting parameters and performance parameters for the problem at hand is:

$$K = N_{perf} + L \cdot P \tag{4.77}$$

the total number of parameters to be estimated by the optimiser becomes:

$$K_{tot} = \frac{K}{2} \cdot (K+3) \tag{4.78}$$

which includes $\mathbf{x},\boldsymbol{\sigma},\boldsymbol{\theta}$.
For a typical case where $N_{perf}=4$, $L=3$, $P=3$, evaluation of (4.78) produces $K_{tot}=104$. Such a large number of parameters to be estimated makes the optimisation task very difficult. Tests have confirmed this. Thus, application of ESs based on correlated mutations has to be discarded.

In conclusion, (μ,λ)-ESs turn out to be the most suitable tool to enhance the real-coded GA for MOPA.

One issue still has to be addressed to comprehensively estimate the technique's suitability, i.e. the balance between GA and ES within the EP optimiser in terms of computational resources and optimal distribution of the iterations. In this respect, two methods can be considered, that identify extreme approaches:

- the GA is used without any ES-based enhancement
- the GA is not used and a number of separate ESs are used for the fault classes. Every ES is run until convergence and the one providing the lowest objective function identifies the faulty engine component.

In between them, various procedures can be considered, that combine both GAs and ESs. Utilisation of the ES strongly influences the EP's performance. Apart from the accuracy of estimation, the way the fault classes evolve depends on the extent to which the ES is used. Fig. 4.15 and 4.22 show how concentration on the faulty engine components is achieved by competition among the fault classes in the GA-based EP. Both plots show that after a phase of strong competition characterising the first generations, a fault class clearly emerges as the winner. An inversion of the trend occurs seldom. It is quite rare that after a significant number of generations a fault class which seems to be doomed to disappear starts getting bigger and bigger, while the currently winning fault class faces a recession phase by loosing an increasing number of individuals.

On the contrary, use of the ES leads to this behaviour. Fig. 4.34-4.35 show two typical plots of the number of strings per class vs. number of generations for the ES-enhanced GA. The influence of the ES, that is here used every 10 GA generations, is evident. The effects of the ES on the curve are the following:

- well-performing fault classes are rewarded. In particular the winning fault class soon shows a steep rate of increase and gains a very large number of representatives
- a small number of fault classes usually survive for a large number of generations with a small number of strings
- most of the curves are non-smooth. The major non-smoothness is originated just after application of the ES
- when the diagnostic problem of identifying the faulty engine components is particularly complex, the curves number of strings per class vs. number of generations may take on an oscillatory shape (fig. 4.35). This means that the fault class identifying the actually faulty components may run the risk of becoming empty before the EP settles down.

From a diagnostic viewpoint, the only concern may be due to a curve like the one shown in 4.35, which may lead to misdiagnosis of the faults.

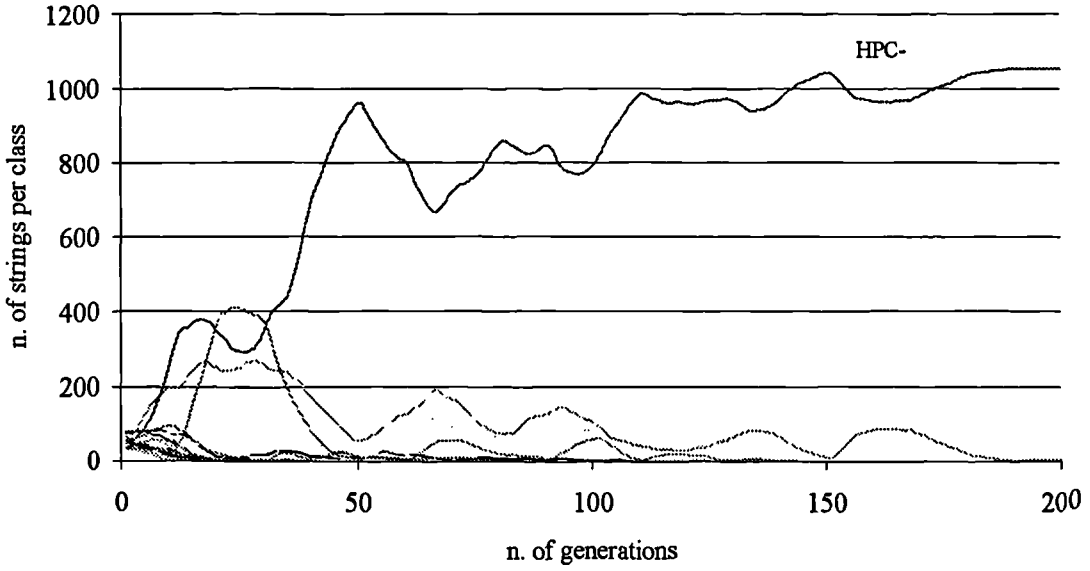


Fig. 4.34: concentration for 3 operating point analysis

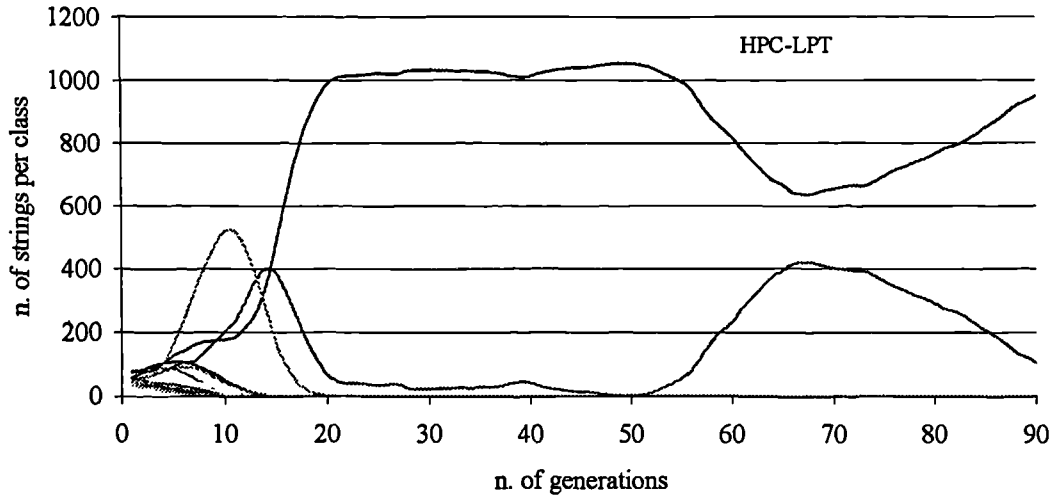


Fig. 4.35: concentration for 2 operating point analysis

The problem can be partially solved by increasing the total number of strings. That has the following effects:

- the computational time involved increases
- fault classes are less likely to die off quickly and hence the chance of missing interesting solutions decreases
- the oscillatory shape of the curves is unlikely to disappear. Increasing the number of strings per fault class enlarges the diversity present in each subpopulation. As the ES is used as a local search method involving just few strings per class, it is quite likely for the worst strings of the best fault class to be worse than the best strings of other fault classes for a large number of generations.

Fig. 4.36 through 4.38 show the results obtained by analysing a diagnostic case with a different number of operating points. The engine is the EJ200, faults have been implanted in the HP compressor and LP turbine. The following set made of 8 measurements is used:

- engine inlet airflow (W_{1A})
- fan outer exit total temperature (T_{13})
- fan inner exit total temperature (T_{12})
- HP compressor exit total pressure and temperature (P_3, T_3)
- LP turbine exit total pressure (P_5)
- HP spool rotational speed (N_H)
- LP spool rotational speed (N_L).

T_3 and P_5 are affected by a 1% bias. Rotational speeds are not supposed to be biased. The three nominal values of fuel flow are: 800, 850, 900 g/s.

parameters	actual (%)	predicted (%)
$\Delta\eta_{FANOUT}$	0.	0.
$\Delta\Gamma_{FAN}$	0.	-0.65
$\Delta\eta_{FANIN}$	0.	-2.87
$\Delta\Gamma_{HPC}$	-3.	0.
$\Delta\eta_{HPC}$	-3.	0.
$\Delta\Gamma_{HPT}$	0.	0.
$\Delta\eta_{HPT}$	0.	0.
$\Delta\Gamma_{LPT}$	3.	3.
$\Delta\eta_{LPT}$	-1.	-2.03
ΔC_D	0.	0.
		RMS = 1.66
operating point 1		
W_{FE} (g/s)	767.3	761.2
P_{AMB} (kN/m^2)	101.311	101.622
T_{AMB} (K)	287.2	288.1

Fig. 4.36: results with one operating point

parameters	actual (%)	predicted (%)
$\Delta\eta_{FANOUT}$	0.	0.
$\Delta\Gamma_{FAN}$	0.	0.
$\Delta\eta_{FANIN}$	0.	0.
$\Delta\Gamma_{HPC}$	-3.	-2.93
$\Delta\eta_{HPC}$	-3.	-1.40
$\Delta\Gamma_{HPT}$	0.	0.
$\Delta\eta_{HPT}$	0.	0.
$\Delta\Gamma_{LPT}$	3.	2.10
$\Delta\eta_{LPT}$	-1.	-2.12
ΔC_D	0.	0.
		RMS = 0.68
operating point 1		
W_{FE} (g/s)	833.1	833.1
P_{AMB} (kN/m^2)	101.289	101.196
T_{AMB} (K)	288.3	288.1
operating point 2		
W_{FE} (g/s)	884.8	883.7
P_{AMB} (kN/m^2)	101.337	101.291
T_{AMB} (K)	288.8	288.8

Fig. 4.37: results with two operating points

parameters	actual (%)	predicted (%)
$\Delta\eta_{FANOUT}$	0.	0.
$\Delta\Gamma_{FAN}$	0.	0.
$\Delta\eta_{FANIN}$	0.	0.
$\Delta\Gamma_{HPC}$	-3.	-2.84
$\Delta\eta_{HPC}$	-3.	-2.14
$\Delta\Gamma_{HPT}$	0.	0.
$\Delta\eta_{HPT}$	0.	0.
$\Delta\Gamma_{LPT}$	3.	2.45
$\Delta\eta_{LPT}$	-1.	-2.25
ΔC_D	0.	0.
		RMS = 0.51
operating point 1		
W_{FE} (g/s)	766.8	768.1
P_{AMB} (kN/m^2)	101.310	101.260
T_{AMB} (K)	288.3	288.4
operating point 2		
W_{FE} (g/s)	814.5	815.2
P_{AMB} (kN/m^2)	101.284	101.109
T_{AMB} (K)	287.5	287.5
operating point 3		
W_{FE} (g/s)	862.9	864.1
P_{AMB} (kN/m^2)	101.319	101.104
T_{AMB} (K)	288.0	288.3

Fig. 4.38: results with three operating points

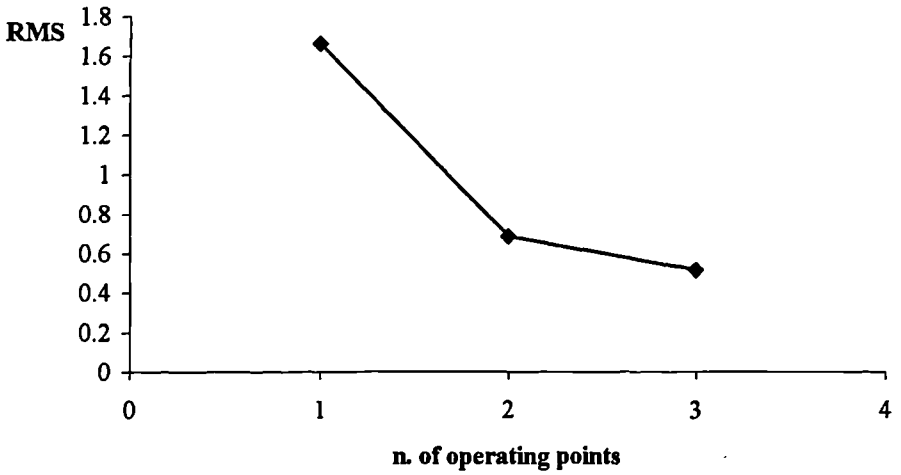


Fig. 4.39: RMS vs. number of operating points

A comparison among the three diagnostic answers allows to draw interesting conclusions. Increasing the number of operating points increases the answer's accuracy. In particular utilisation of a single operating point leads to incorrect identification of the faulty engine components, whereas using more operating points enables to isolate the

faults correctly. Evaluation of (4.76) for the case at hand produces a value of 0.86 for the single operating point analysis. The method's unsuitability to handle the diagnostic problem, suggested by the approximate treatment leading to eq. (4.76), is confirmed by the results. Fig. 4.39 shows how the RMS is depending on the number of operating points. Again, the plot is well consistent with fig. 4.23, showing the relative redundancy index vs. the number of operating points.
It is worth to point out that in the single operating point case the optimiser finds it difficult to precisely accommodate the biased fuel flow measurement as well. However in all the three cases the biases affecting the monitoring measurements T_3 and P_5 are correctly isolated.
Fig. 4.40 shows the typical convergence curve of the EP-based optimiser for MOPA. The steps in the objective function values are easily detectable in correspondence of the application of the ES.

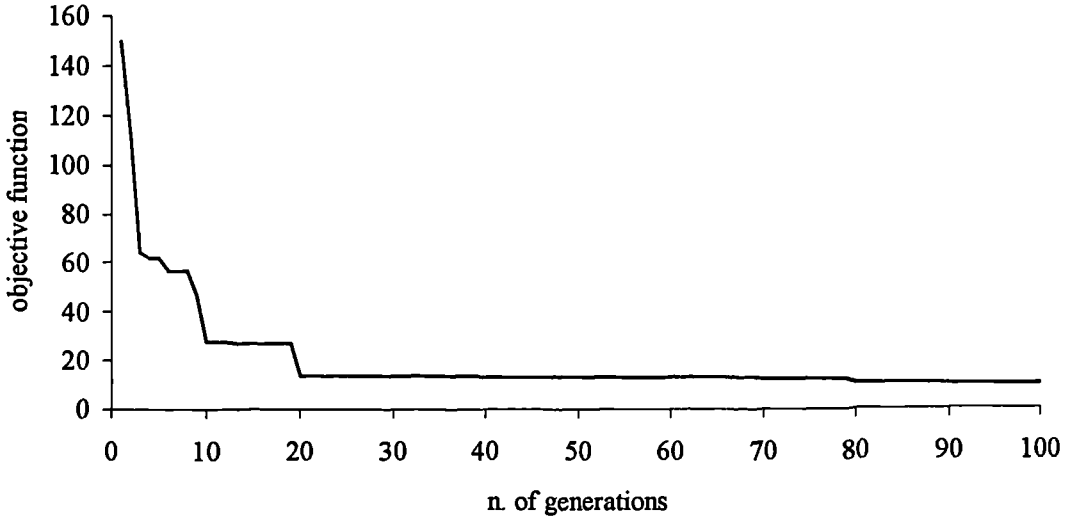


Fig. 4.40: convergence curve for ES-GA-based EP for MOPA

The following remarks can be made on the ES-enhanced EP:

1. optimisation of the MOPA function (4.72)-(4.73) is fostered. Convergence times are shortened with respect to the GA-based EP and the objective function's final value is most of the times close to or even smaller than the target one
2. convergence times become longer when the number of fault classes is larger (e.g. three-spool engines)
3. utilisation of a large number of strings is required to reduce the probability of disappearance of the fault class sought for. In turn, this increases the number of objective function evaluations necessary for convergence and hence the convergence time
4. point 1 on one side and points 2 and 3 on the other side balance out and the computational time required by the ES-enhanced EP for MOPA is comparable with the computational time required by the ES-free EP for single operating point analysis

5. use of the ES for single operating point analysis is suggested because optimisation is made easier. Moreover simulations have shown that the computational time is not increased because of point 4 and the faster evaluation of the objective function for one operating condition.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Summary of thesis' objectives

The present work went through four main stages, as described below (see also section 1.4).

- *Review of diagnostic techniques.* Various types of diagnostic tasks are required in order to efficiently and safely operate a gas turbine. Performance analysis is used in different forms at various stages of a gas turbine life. The various diagnostic tasks can be accomplished by means of different techniques. The first part of the present work aimed at analysing the existing techniques for the various diagnostic problems. Whenever possible, a critical assessment of the methods was provided. New methods have been developed and proposed to cope with the limitations of classic approaches. A wide range of advanced techniques and their suitability for the problems at hand have been analysed. This phase of the project is to be regarded as particularly important, as it provided information, knowledge and ideas for the successful development of the diagnostic method proposed.
- *Application of neural networks to gas turbine diagnostics.* Among various promising techniques, NNs show interesting features that make them especially suitable to tackle difficult diagnostic problems. Therefore, after a thorough review of the work available on the subject in the public literature, two developments are described, which envisage a massive use of NNs for diagnostics. Whereas in the first study NNs were used to carry out comprehensive diagnostics without use of model-based tools, the second study focused on application of NNs to solve the measurement validation problem.
- *Fault diagnosis method for development engines.* The heart of the project was the development of a diagnostic method suitable for well-instrumented engines, typically analysed in development test bed. All measurement uncertainty effects are taken into account. Major target of the method is concentration on the fault-affected engine components. The method is based on optimisation of a tailored objective function. The minimisation is effected by means of an evolution program based on genetic algorithms.
- *Fault diagnosis method for pass-off tests.* After development and testing of the diagnostic method for well-instrumented engines, the technique is expanded to deal with poorly instrumented engines as well. The objective function is modified to perform multiple operating point analysis through an evolution program based on both genetic algorithms and evolution strategies.

In the sequel the results obtained are commented and recommendations provided

5.2 Promising diagnostic techniques

A large number of diagnostic techniques have been reviewed and analysed. The following recommendations are given for further development of gas turbine diagnostics.

- Conventional Kalman filter-based techniques have long been the backbone of most gas turbine diagnostic methods. Poor statistical knowledge on the probability of fault occurrence and evolution, need for tuning, "smearing" effect, divergence due to modelling errors and linearisation are common problems that affect most of the Kalman filter-based methods (see section 2.2.1). Even though Kalman filtering is still the most widespread estimation technique, new approaches have to be investigated and tested in the various areas of gas turbine diagnostics in order to establish new standards.
- The German approach to the gas turbine diagnostic problem (section 2.4) is interesting in that it shows some attempts to integrate the KF-based method with new techniques. The two key points of the German work are: need for integration of different methods to suit the particular diagnostic problem at hand and use of Artificial Intelligence-based methods. It is believed that most techniques to be developed in the future for gas turbine diagnostics will show the above-mentioned properties.
- The diagnostic technique based on the eigenstructure assignment method (section 2.5) is strongly recommended for a number of reasons: the capability to handle model errors and then safely use a linearised model is remarkable, the decoupling of residuals from disturbances allows to identify and correct measurement errors and the overall computational burden is limited. It is also worth investigating the suitability of using the Schur method to further reduce the computational complexity after introduction of the performance parameters in the set of quantities to be estimated.
- It is believed that transient performance analysis will be given increasing importance in the near future, also because of improvements in transient modelling. If an accurate and reliable transient performance model is available, a great deal of diagnostic information can be extracted from the sensor suite. The Bayesian technique introduced in section 2.6 and the modification proposed for the real case of different measurement noise levels should be tested with accurate non-linear models. In particular the method's robustness to measurement uncertainty has to be ascertained. Sensible transient measurement time constants have to be taken into account as well.
- The Minimum Model Error estimator (section 2.8) has to be tested for the problem of tracking the temporal evolution of deterioration. The technique's capability to deal with poor knowledge about the fault dynamics should represent a remarkable advantage. In particular a computational feasibility study has to be made in order to ascertain whether the method given by (2.176)-(2.181) is practical.
- The Maximum Likelihood estimation (section 2.9) has to be developed and tested for comparison with the results obtained by the MME estimator. Although performance prediction of non-linear estimators is particularly difficult, the MME estimator is believed to be able to produce better results due to its inherent features. Nonetheless, a comparison test with the ML estimator would be necessary.
- A straightforward move from standard to non-linear Kalman filtering (section 2.7) can easily be made by implementing the Iterated Extended Kalman filter. A comparison of

performance between the two filters can definitely be useful. However, it is believed that more significant gains can be achieved by using optimisation techniques, as shown in chapter 4. Better accuracy and flexibility is likely to be achievable by optimisation-based procedures, even though the corresponding computational burden usually increases.

5.3 Suitability of NNs as a diagnostic tool for gas turbines

After thorough analysis of the public literature on the subject, a wide range of NN applications have been developed and tested. As stated earlier, the large number of NN architectures and the large number of diagnostic objectives make it somewhat difficult to select the best use of NNs for diagnostics.
Two main points uniquely characterise the two studies carried out on NNs:

- the approaches are based on intensive use of computing power. Sometimes NNs are used because they can perform a task with less computational burden than classic techniques. In this case, though, the diagnostic tasks are difficult to accomplish and the use of a limited computing power has been considered as a secondary requirement. Development of the various neural techniques aimed at testing the extreme performance achievable
- real world effects, mainly related to measurement uncertainty, are taken into account as much as possible in an attempt to assess NN performance in a realistic environment. It is worthwhile underlining that:
  1. the models used for generating data are accurate and fully non-linear (especially the one used for the work on data validation)
  2. real noise levels have been used
  3. all possible sources of measurement uncertainty have been accounted for (noise and biases affecting all measured parameters)
  4. as far as fault diagnosis is concerned, such a large number of biased measurements as assumed in the sensor validation work is rather unrealistic, especially for test bed analysis.

In the light of the published studies and the results of the works described, the following comments can be made as to the applicability of NN techniques to gas turbine diagnostics:

- even though performance parameters could be calculated with a good accuracy by means of NNs, a model-based method is believed to be more suitable because of the greater flexibility provided. For instance, change of the operating point to be analysed would require minor modifications to the model-based method, whereas a large number of NNs would have to be trained anew. Furthermore, model-based methods usually can be given statistical interpretation and as such can be properly tailored to the problem at hand
- the main hurdle of gas turbine diagnostic is the simultaneous presence of engine and sensor faults: measurements used to estimate engine component faults may be themselves affected by biases. NNs offer a peculiar way to somehow separate the two problems of bias and performance parameter estimation. From an analytical point

of view, the link between performance parameters and biases is due to the coupling of the non-linear equations (see eq. (3.44) and (3.45)). The self-supervised nature of AANN training allows overcoming the problem, provided that the location of engine faults is known. If measurement validation is carried out when no information is available yet as to the engine component faults, then a bank of NNs is necessary. Need for such a large number of nets trained with data covering just a small area of the operational envelope means that the system is little flexible. However, if this lack of flexibility is accepted, effective SFDI can be accomplished by means of careful NN design

- when measurement validation has to be performed in presence of engine component faults, NNs are not reliable enough to accommodate the isolated biased measurements
- NNs can help reduce the noise scatter significantly
- NNs are very effective in tracking trends and they should be preferred to conventional methods for analysis of time varying processes. In general, although they can be useful for analysis of transient behaviour, where an accurate dynamic model is available, NNs can be successfully applied to track trends of processes whose dynamics is poorly modelled or even utterly unknown (e.g. temporal evolution of faults analysed at steady state)
- NNs can be regarded as "black box" techniques: they simply learn from examples and it is very difficult to get information as to why a certain diagnostic answer has been arrived at. As a matter of fact, unless the process analysed by NNs is very simple, the hidden neurons can usually be given no physical meaning. Therefore, even providing a level of confidence of the produced diagnostics is unfeasible. From this point of view, model-based and rule-based techniques seem to be more suitable
- eventually, another facet of the problem should not be neglected: a technique is *considered successful when it is fully understood and accepted by the user.* The following remarks have to be made in this respect:
  1. even though NNs can be regarded as rather simple methods, they have to be used with care. In particular, effective utilisation of NNs is possible only when their limitations are known as well. That entails that the user should be quite familiar with the technique
  2. although NNs often perform simple function approximations, their inherent features make them somewhat different from many other well known and widely used techniques. In particular, the fact that they are not model-based can actually be one of the reasons why the user may reject the technique on presumption of lack of physical plausibility.

5.4 Optimisation-based engine and sensor fault diagnosis with comprehensive instrumentation set

The optimisation-based method for performance analysis of development engines definitely represents the bulk of the present work. The novelty of the approach is evident and its effectiveness and accuracy have been tested.

For a complete assessment of the method's performance it is necessary to compare it to typical KF-based techniques. In this respect, the following points have to be made:

- The proposed method is inherently more robust to measurement biases. KF-based techniques estimate engine component and sensor faults by using measurements as input. The estimation problem can therefore be considered highly underdetermined. Occurrence of a relatively large number of biases is very likely to hinder a successful application of the KF-based technique. On the contrary, the ability of the proposed method to concentrate on a limited number of fault-affected performance parameters by using bias-free measurements allows accounting for a larger number of biases. A simple comparison with the performance claimed for any KF-based approach confirms this relevant advantage of the proposed method.
- The optimisation-based method relies on a fully non-linear model. The results achieved are more accurate in term of RMS than the ones usually obtained by means of linearised techniques. Even cases of relatively large deterioration or variations of up to four performance parameters can be dealt with easily.
- The parameters setting the operating condition of the engine are easily corrected to account for both noise and biases. Most of the KF-based methods find it difficult to account for this measurement uncertainty effects.
- Concentration on the faulty engine components is easily effected, whereas the "smearing" effect affects most KF-based approaches. This outstanding advantage of the proposed method produces a clear indication of the location and an accurate quantification of the fault.
- The optimisation-based method is robust with respect to small modelling and measurement errors. On the contrary, the KF is intrinsically non-robust to discrepancies from the assumed model.
- The proposed method requires minimal statistical information. Measurement noise levels and maximum possible number of faulty engine components and sensors are easily available. The result's accuracy is a weak function of the initial distribution of strings among the various fault classes. Conversely, tuning of the KF is likely to strongly bias the diagnostic answer.

The method evidently shows many novel features. In particular:

- Genetic algorithms and evolution strategies are certainly a new entry in the field of gas turbine diagnostics.
- Concentration is accomplished by using the genetic operators in a problem-specific way. The peculiar use of selection, crossover and mutation allows isolating sensor faults.
- The concept of analytical redundancy is used in a peculiar fashion in an optimisation-based framework.
- Use of the non-quadratic objective function, possible because of the optimisation capability of evolution programs, allows coping with small modelling and measurement errors.
- The way environment and power setting parameters are dealt with is novel as well.

Notwithstanding the remarkable accuracy and novelty of the proposed method the following drawbacks cannot be neglected:

- The technique is definitely much more burdensome than classic estimation methods from a computational point of view. In this respect two remarks have to be made. Firstly, the requirement for short computing times was regarded since the beginning of the project as a secondary one. Accuracy of diagnostics was correctly considered as a more important target. Secondly, the ever-increasing availability of computer power at low cost allows to envisage that what is reckoned as computationally heavy now may be an easy job in the near future. Even during the development of the present project, the platform used for all the simulations has soon become old-fashioned. Nowadays the market already offers newer and relatively cheap machines that outperform what three years ago could be regarded as the top performance platform.
- The diagnostic method, albeit supported by a theoretical soundness, does not provide information on the level of confidence of the diagnostic answer produced. On the contrary, classic maximum likelihood methods applied to linear systems allow calculating the probability that the proposed solution is the one sought for. Two observations can be made in this respect: firstly, the incapability of assigning a level of confidence to the proposed solution directly stems from the attempt to limit as much as possible approximations (e.g. linearisations); secondly, although a statistically meaningful level of confidence cannot be attached to the proposed diagnostic answer, experience easily shows what rough value the objective function has to reach for the optimisation to be regarded as successful (given a certain number of measurements used for monitoring the engine performance).
- The statistical input required by the method is really minimal with respect to most of the other estimation techniques. However, care must be paid in assigning the number of strings to the various fault classes. Even though the rule of assignment of the number of strings for different fault classes can easily be established by trial and error and the accuracy is not a strong function of the rule of assignment itself, awareness of this issue is necessary for a correct utilisation of the technique.

Further work should be done according to the following recommendations:

- The advantages gained by application of ESs to the GA-based optimisation should be quantified in terms of time to convergence. A large number of runs should be made in order to establish statistically significant figures on the matter. From the simulations already carried out, the advantage is supposed to be relevant.
- Some tuning studies should be made to optimise the choice of the number of ES generations with respect to the GA generations and the initial size of the ES strategy parameters. An attempt should be made to linearly reduce the size of the initial strategy parameters whilst the convergence of the GA population proceeds.
- Testing should be made with real data and performance simulation models tailored for analysis of those data. Although attention has been paid to the issue of model uncertainty, testing with real data certainly represents a benchmark for the method's validation.
- Testing should be made with other engines, especially from the three-spool family (e.g. Trent), in an attempt to establish how practical the method is in terms of the required computing power.

5.5 Optimisation-based engine and sensor fault diagnosis with poor instrumentation set

The optimisation-based method was purposely developed to deal with performance analysis of well-instrumented engines. The encouraging results achieved suggested that an expansion of the technique to handle the case of poorly instrumented engines was to be pursued. From conceptual point of view, the multiple operating point analysis certainly is a natural offspring of the single operating point analysis. However, serious problems of convergence were soon encountered and called for modifications of the optimisation procedure that initially was only based on GAs. Application of the ESs to enhance the optimiser has shown the feasibility of the optimisation at hand.

Most of the remarks made about single operating point analysis are applicable here as well. Specifically, the following observations pertain to the optimisation-based MOPA:

- A large number of simulations have to be run in order to establish statistically significant figures as to the effect of the number of operating points on the diagnostic accuracy.
- Test has to be made to evaluate how far the operating points can be from each other for the diagnostics to be meaningful. Again, utilisation of real data and actual maps (possibly distorted by component faults) is mandatory.
- Possible advantages of the multi-objective optimisation method (Goldberg, 1989) have to be thoroughly investigated. It is believed that major benefits could be gained in the following areas: firstly, application of a Pareto-based approach could foster the optimiser's convergence, especially in terms of isolation of the faulty components; secondly, utilisation of multi-objective optimisation could allow to weigh the information coming from different operating points. The proposed solutions would clearly show which operating points have contributed most to the diagnostic answer.

REFERENCES

- Alag, G. S., Gilyard, G. B., 1990: "A proposed Kalman filter algorithm for estimation of unmeasured output variables for an F100 turbofan engine", NASA Technical Memorandum TM 4234
- Almeida, L.B., and Neto, J.P., 1990: "Recurrent backpropagation and Hopfield networks", contained in Neurocomputing, Springer-Verlag editor
- Antonisse, J., 1989: "A new interpretation of schema notation that overturns the binary encoding constraint", Proceedings of the Third International Conference on Genetic Algorithms
- Anuzis, P., 1998: "Design for reliability", handouts of a Rolls-Royce seminar held in Cranfield in November
- Baker, J.E., 1987:"Reducing bias and inefficiency in the selection algorithm", Proceeding of the Second International Conference on Genetic Algorithms
- Baker, J.E., 1989: "Adaptive selection methods for genetic algorithms", International Conference on Genetic Algorithms
- Baldi, P., 1995: "Gradient descent learning algorithm overview: a general dynamical systems' perspective", IEEE Transactions on Neural Networks, vol. 6, n. 1, January
- Barwell, M. J., 1987: "COMPASS : ground-based engine monitoring program for general application", SAE- 87/1734, SAE Aerospace Technology Conference, October 5-8/Long Beach, CA - Affiliation: Rolls Royce
- Baumgartner, E.T., 1991: "Alternative smoothing algorithms for on-line estimation problems", paper AIAA-91-0194, 29th Aerospace Science Meeting, January 7-10, Reno, NV
- Baumgartner, E.T., Walker, B.K., 1990: "Nonlinear smoothing for the continuous-discrete problem with application to rocket health monitoring", paper AIAA-90-3405-CP
- Bickmore, T.W., 1992: "A probabilistic approach to sensor data validation", paper AIAA-92-3163, AIAA/SAE/ASME/ASEE 28th Joint Proplusion Conference and Exhibition, July 6-8, Nashville, TN
- Bryson, A. E. Jr., Ho, Y., 1975: "Applied optimal control : optimization, estimation, and control", revised printing, Hemisphere Publishing Corporation, Washington
- Carr, H.C., and Cowley, P. H., 1995: "Application of Neural Networks to aero-engine vibration monitoring", EPF 95-26, AIDAA/AAAF/DGLR/ RAeS 5th European Forum, Pisa, Italy
- Chen, J., Patton, R.J., and Liu, G.P., 1994: "Design of optimal residuals for detecting sensor faults using multi-objective optimization and genetic algorithms", AIAA 94-3581-CP
- Cifaldi, M.L. and Chokani, N., 1998: "Engine monitoring using neural networks", 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, Cleveland, CA, July 12-15
- Cohen, H., Rogers, G.F.C., and Saravanamutto, H.I.H., 1986: "Gas turbine theory", 3rd edition, Longman Scientific & Technical editor
- Consumi, M., 1996: "Engine condition monitoring tramite identificazione Bayesiana", thesis of Aeronautical Engineering, University of Pisa, Italy

- Consumi, M. and D'Agostino, L., 1997: "Monitoring and fault diagnosis of a turbojet by Bayesian approach", ISABE 97-7146, 13th ISABE Conference, 8-12 September, Chattanooga, TN
- Consumi, M. and D'Agostino, L., 1998: "A statistical inference approach to gas path analysis of a turbofan", paper AIAA 98-3551, 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 13-15, Cleveland, OH
- Curnock, B., 1995: "RB199 performance diagnostic system: PADS. Review of data produced by the field trial", Rolls-Royce internal technical report, ref. DNS 31499
- De Jong, K, 1989: "Genetic algorithms: a 10 year perspective", International Conference on Genetic Algorithms
- De Laat, J. C., Merrill, W. C., 1990: "Advanced detection, isolation and accomodation of sensor faulires in turbofan engines", NASA Technical Paper TP 2925
- Doel, D.L., La Pierre, L.R., 1989: "Diagnostic expert systems for gas turbine engines - Status and prospects", paper AIAA-89-2585, AIAA/ASME/SAE/ASEE 25th Joint Propulsion Conference, Monterey, CA
- Doel, D.L.,1994: "TEMPER4: gas-path analysis tool for commercial jet engines", ASME/92-GT-315, Transactions of ASME, Journal of engineering for gas turbines and power, Vol. 116, No. 1, January
- Doel, D. L., 1994: "An assessment of weighted-least-squares-based gas path analysis", ASME/93-GT-119, Transactions of ASME. Journal of engineering for gas turbines and power, Vol. 116, No. 2, April
- Duponcel, J.P., Loisy, J.,and Carrillo, R., 1992: "Steady and transient performance calculation method for prediction, analysis and identification", Advisory Group for Aerospace Research & Development, Neuilly-sur-Seine
- Dyson, R.J.E, 1984: "CF6-80 condition monitoring -- the engine manufacturer's involvement in data acquisition and analysis", AIAA-84-1412, Paper presented at: AIAA/SAE/ASME 20th joint propulsion conference. - Affiliation: Aircraft Engine Business Group
- English, L.A., 1995:"Application of gas path analysis, gas path debris monitoring and expert system technology to the Allison T56 Turboprop engine", MSc thesis, Thermal Power Department, School of Mechanical Engineering, Cranfield University
- Escher, P., 1995: "Pythia: an object-oriented gas path analysis computer program for general applications", PhD Thesis, School of Mechanical Engineering, Cranfield University
- España, D. M., 1993: "On the optimization algorithm for adaptive performance optimization of turbofan engines", AIAA-93-1823, AIAA/SAE/ASME/ASEE 29th Joint Propulsion Conference and Exhibit, Monterey, CA, June 28-30
- España, M. D., Gilyard, G. B., 1993: "On the estimation algorithm used in adaptive performance optimization of turbofan engines", NASA Technical Memorandum TM-4551
- Eustace, R., 1993: "Neural Network fault diagnosis of a turbofan engine", XI ISABE 93-7091, Tokyo, Japan, September 19-24, pp.937-946
- Eustace, R., and Merrington, G., 1995: "Fault diagnosis of fleet engines using Neural Networks", ISABE 95-7085, pp.926-936

- Fiedler, K. and Lunderstaedt, R., 1985: "Zur systemtheoretischen Diagnose von Strahltriebwerken", Automatisierungstechnik, at 33, pp. 272-279, 313-317
- Gelb, A., 1974: "Applied optimal estimation", M.I.T. Press, Cambridge, MA
- Gertler, J., and Luo, Q., 1989: "Robust isolable models for failure diagnosis", AIChE Journal, November, vol. 35, n. 11
- Goldberg, D.E., and Richardson, J, 1987:"Genetic algorithms with sharing for multimodal function optimisation", Proceedings of the Second International Conference on Genetic Algorithms
- Goldberg, D.E., 1989: "Genetic algorithms in search, optimization and machine learning", Addison Wesley Publishing Company Inc.
- Goldberg, D. E., 1990: "Real-coded genetic algorithms, virtual alphabets and blocking", IlliGAL report 90001, September, University of Illinois at Urbana-Champaign, IL
- Grainger Allen, P.W., 1998: "Engine performance program user manual", RRAP customer deck documentation
- Grefenstette, J.J., and Baker, J.E., 1989: "How genetic algorithms work: a critical look at implicit parallelism", International Conference on Genetic Algorithms
- Grefenstette, J.J., and Fitzpatrick, J.M., 1990: "Genetic search with approximate function evaluations", International Conference on Genetic Algorithms
- Grove, R.D., Bowles, R.L., Mayhey, S.C., 1972: "A procedure for estimating stability and control parameters from flight test data by using maximum likelihood methods employing a real time digital system", NASA TN D-6735
- Guo, T.H., Saus, J., Lin, C.F., and Ge, J.H., 1996: "Sensor validation for turbofan engines using an autoassociative neural network", AIAA 96-3926, AIAA Guidance, Navigation and Control Conference, July 29-31, San Diego, CA
- Hassoun, M. H., 1995: "Fundamentals of artificial neural networks", MIT Press, London
- Haupt, G.T., Kasdin, N.J., Keiser, G.M., and Parkinson, B.W., 1995: " An optimal recursive iterative algorithm for discrete nonlinear least squares estimation", AIAA-95-3218-CP
- Haykin, S., 1994: "Neural Networks: a comprehensive foundation", MacMillan College Publishing Company, New York
- Hillemann, T. and Hopermann, H., 1992: "Wissenbasierte Sensorfehlerdetektion an Gasturbinen unter Verwendung unscharfer Verfahren", Workshop on Fuzzy Control of Gesellschaft fuer Automatisierungstechnik, UA, University of Dortmund
- Hollingsworth, D., 1994: "Monitoring of civil aero-engines", Rolls-Royce Magazine, March
- Huber, P.J., 1981:"Robust statistics", John Wiley & Sons
- Janikow, C.Z. and Michalewicz, Z., 1991: "An experimental comparison of binary and floating point representations in Genetic Algorithms", Proceedings of the Fourth International Conference on Genetic Algorithms
- Jazwinski, A.H., 1970: "Stochastic process and filtering theory", Academic Press, New York
- Junk, R. and Lunderstaedt, R., 1995: "Model optimization of a gas turbine with fault affected reference measurements by neuronal networks for state and sensor

diagnosis", AAIDA/AAAF/DGLR/RAeS 5th European Propulsion Forum, Pisa, Italy

- Kanelopulos, K., Stamatis, A. and Mathioudakis, K., 1997: "Incorporating neural networks into gas turbine performance diagnostics", paper ASME 97-GT-35, International Gas Turbine & Aeroengine Congress & Exhibition, June 2-5 Orlando, FL
- Kelly, R.W., Dadd, G.J., and Martin, J.R., 1996: "Final report on performance optimization for multivariable aircraft engines", DRA internal report
- Kramer, M.A., 1991: "Non-linear principal component analysis using autoassociative networks", AIChE Journal, vol.37, n. 2, pp. 233-243
- Kramer, M.A., 1992: "Autoassociative neural networks", Computers & Chemical Engineering, vol. 16, n. 4, pp. 313-328
- Krishnakumar, K.,1993: "Genetic algorithms: a robust optimization tool", paper AIAA-93-0315, 31st Aerospace Sciences Meeting and Exhibit, January 11-14, Reno, NV
- Katafygiotis, L.S., 1991: "Treatment of model uncertainties in structural dynamics", EERL 91-01, Earthquake Engineering Research Laboratory, California Institute of technology, Pasadena, CA
- Kerr, L. J., Nemec, T. S., Gallops, G.W., 1991: "Real time estimation of gas turbine engine damage using a control-based Kalman filter algorithm", ASME 91-GT-216, Transactions of ASME, Journal of engineering for gas turbines and power
- Lane, R.J., and Behenna, J., 1991: "EJ200: the engine for the new European Fighter Aircraft", 90-GT-119, Journal of Engineering for Gas Turbines and Power, Transactions of the ASME, January, vol. 113
- Launer, R.L. and Wilkinson, G.N., 1979: "Robustness in statistics", Academic Press, New York
- Lee, Y.H., and Singh, R., 1996: "Health monitoring of turbine engine gas path components and measurement instruments", paper ASME 96-GT-242, International Gas Turbine and Aeroengine Congress and Exhibition, Birmingham, UK, June 10-13
- Liu, G.P. and Patton, R.J., 1996: "Robust parametric eigenstructure assignment", AIAA 96-3908, AIAA Guidance Navigation and Control Conference, July 29-31, San Diego, CA
- Lunderstaedt, R. and Fiedler, K., 1988: "Gas path modeling, diagnosis and sensor fault detection", AGARD Conference Preprint Nr 448, Quebec, Canada
- Lunderstaedt, R., 1988: "Zur Kompensation der systematishen Sensorfehler", Automatisierungstechnik, at 36, pp. 282-289
- Lunderstaedt, R., 1990: "Zur Elimination von Sensorfehlern und Beseitigung von Modelldefekten", Automatisierungstechnik, at 38, pp. 223-230
- Lunderstaedt, R. and Hillemann, T., 1992: "Knowledge-based sensor fault detection for gas turbines under consideration of model based methods", Preprints IFAC/IFIP/IMACS, Symposium on Artificial Intelligence in Real Time Control, Delft
- Lunderstaedt, R. and Hillemann, T., 1993: "Sensor fault detection of gas turbines with knowledge-based methods", Preprints of the 12th IFAC Congress, Sidney, Australia
- Lunderstaedt, R., and Junk, R., 1997: "Application of the gas path analysis for the non-stationary operation of a jet engine", ISABE 97-7062, 13th ISABE Conference, 8-12 September, Chattanooga, TN

- Luppold, R. H., Roman, J. R., Gallops, G. W., Kerr, L. J., 1989: "Estimating in-flight engine performance variations using Kalman filter concepts", AIAA-89-2584, AIAA/ASME/SAE/ASEE 25th Joint Propulsion Conference, Monterey, CA, July 10-12
- Mah, R. S. H., and Tamhane, A. C., 1982: "Detection of gross errors in process data", AIChE Journal, September, vol. 28, n.5
- Mah, R.S.H., 1990: "Chemical process structures and information flows", ed. Butterworths
- Maine, T. A., Gilyard, G. B., Lambert, H. H., 1990: "A preliminary evaluation of an F100 engine parameter estimation process using flight data", NASA Technical Memorandum TM-4216
- Mathioudakis, K. and Stamatis, A., 1994: "Compressor fault identification from overall performance data based on adaptive stage stacking", Journal of Engineering for Gas Turbines and Power, vol 1161, January, pp. 156-164
- Mattern, D. L., Jaw, L. C., Guo, T. H. , Graham, R., and McCoy, W., 1997: "Simulation of an engine sensor validation scheme using an autoassociative neural network", AIAA 97-2902, 33rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 6-9, Seattle, WA
- Mattern, D. L., Jaw, L. C., Guo, T. H. , Graham, R., and McCoy, W., 1998: "Using neural networks for sensor validation", paper AIAA 98-3547, 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 13-15, Cleveland, OH
- Mattingly, J.D., 1996: "Elements of gas turbine propulsion", McGraw Hill
- Michalewicz, Z. and Janikow, C. Z., 1991: "Handling constraints in genetic algorithms", Proceedings of the Fourth International conference on Genetic Algorithms, pp. 151-157
- Michalewicz, Z., 1996: "Genetic algorithms + data structures = evolution programs", Springer Verlag, 3rd Edition
- Moller, J.C., Litt, J.S. and Guo, T.H., 1998: "Neural network-based sensor validation for turboshaft engines", paper AIAA 98-3605, 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, Cleveland, CA, July 12-15
- Napolitano, M., Neppach, C. D., Van Casdorph, A., Naylor, S., Innocenti, M., and Bini, F., 1994: " Sensor failure detection, identification and accomodation using on-line neural architectures", AIAA 94-3598, AIAA Guidance, Navigation and Control Conference, August 1-3, Scottsdale, Arizona
- Napolitano, M., Windon, D., Casanova, J., and Innocenti, M., 1996, "A comparison between Kalman filter and Neural Network approaches for sensor validation", paper AIAA 96-3894, AIAA Guidance Navigation and Control Conference, July 29-31, San Diego, CA
- Narducci, R., Grossman, B., Valorani, M., Dadone, A., and Haftka, R.T, 1995: "Optimization methods for non-smooth or noisy objective functions in fluid design problems", AIAA 95-1648-CP
- Nayer, P.R., 1994: "Evaluation of neural networks for engine performance test data analysis - Status report", Rolls-Royce Performance Technical Report PTR 90644

- Neppach, C. D., and Van Casdorph, 1995: "Sensor failure detection, identification and accomodation in a system without sensor redundancy", AIAA 95-0011, 33rd Aerospace Sciences Meeting & Exhibit, January 9-12, Reno, NE
- Palmer, C.A., 1998: "Combining Bayesian belief networks with gas path analysis for test cell diagnostics and overhaul", ASME 98-GT-168, International Gas Turbine & Aeroengine Congress and Exhibition, Stockholm, Sweden, June 2-5
- Patton, R.J. and Chen, J., 1991: "Robust fault detection of jet engine sensor systems using eigenstructure assignment", AIAA 91-2797-CP
- Patton, R.J. and Chen, J., 1995: "Neural network based fault diagnosis for nonlinear dynamic systems", paper AIAA-95-3219-CP
- Pearlmutter, B.A., 1995: "Gradient calculations for dynamic recurrent neural networks: a survey", IEEE Transactions on neural networks, vol. 6, n. 5, September
- Pineda, F.J., 1988: "Generalization of backpropagation to recurrent and higher order neural networks", in Neural Information Processing Systems, D.Z. Anderson editor
- Powell, D. and Skolnick, M. M., Using genetic algorithms in engineering design optimisation with non-linear constraints", Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 424-430
- Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P., 1992: "Numerical recipes in Fortran - The art of scientific computing", 2nd edition, Cambridge University Press
- Provost, M.J., 1988: "COMPASS: a generalized ground based monitoring system", AGARD Conference Preprint Nr 448, Quebec, Canada
- Provost, M.J., 1995: "The use of optimal estimation techniques in the analysis of gas turbines", PhD thesis, Thermal Power Department, School of Mechanical Engineering, Cranfield University
- Provost, M. J., Singh, R., 1995: "Gas path analysis: preparing for success", AAIDA/AAAF/DGLR/RAeS 5th European Propulsion Forum, Pisa, Italy
- Richardson, J. , Palmer, M., Liepins, G. and Hillard, M., 1989: "Some guidelines for genetic algorithms with penalty functions", Proceedings of the Third International Conference on Genetic Algorithms, pp. 191-197
- Sammarco, M., 1988: "Sviluppo di un metodo di diagnosi dei guasti di una turbina a gas", thesis of Aeronautical Engineering, University of Pisa, Italy
- Saravanamuttoo, H.I.H., et al., 1990: "Recommended practices for measurement of gas path pressures and temperatures for performance assessment of aircraft turbine engines and components", AGARD-AR-245
- Schaffer, J.D., 1990: "Multiple objective optimization with vector evaluated genetic algorithms", Interational Conference on Genetic Algorithms
- Schoenauer, M. and Xanthakis, S., 1993: Constrained GA optimisation", Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 573-580
- Singh, R. and Escher, P., 1995: "Gas turbine diagnostics and availability", Industrial and Power Gas Turbine Operation & Maintainance Conference, London, UK
- Singh, R., Zedda, M. and Gulati, A., 1999: "Advanded aircraft engine condition monitoring", Aircraft Technology Engine Yearbook 1999-2000
- Spiegel, M.R., 1972: "Theory and problems of statistics in SI units", Mc Graw Hill, New York

- Stamatis, A., Mathioudakis, K., Berios, G., Papaliou, K., 1989: "Jet engine fault detection with differential gas path analysis at discrete operating points", ISABE 89-7133
- Stamatis, A., Papaliou, K. D., 1988: "Discrete operating condition gas path analysis", AGARD Conference Preprint Nr 448, Quebec, Canada
- Singh, R., and Hemeury, P., 1996: "The simulation and monitoring of performance degradation as an aid to minimizing aero-engine operating costs", 5th Aero-engine Overhaul and Maintenance, London, UK, 24-25 October
- Symard, P.Y., Ottaway, M.B., and Ballard, D.H., 1988: "Analysis of recurrent backpropagation", Technical Report 253, Computer Science, University of Rochester, June
- Tang, G., Yates, C.L. and Chen, D, 1998: "Comparative study of two neural networks applied to jet engine fault diagnosis", paper AIAA 98-3549, 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 13-15, Cleveland, OH
- Tang, G., Yates, C.L. and Chen, D, 1999: "A practical intelligent system for condition monitoring and fault diagnosis of jet engines", paper AIAA 99-2533, 35th AIAA/SAE/ASME/ASEE Joint Propulsion Conference and Exhibit, June 20-24, Los Angeles, CA
- Torella, G., and Lombardo, G., 1995: "Neural Networks for diagnostics and trouble shooting of aero-engines", EPF 95-03, AIDAA/AAAF/DGLR/RAeS 5th European Forum, Pisa, Italy
- Torella, G., 1997: "Expert systems and neural networks for fault isolation in gas turbines", ISABE 97-7148,13th ISABE Conference, 8-12 September, Chattanooga, TN
- Tulpule, S, and Galinaitis, W. S., 1988: "Health monitoring system for the SSME-Fault detection algorithms", AIAA 90-1988, AIAA/SAE/ASME/ASEE 26th Joint Propulsion Conference, July 16-18, Orlando, FL
- Urban, L.A., 1972: "Gas path analysis applied to turbine engine condition monitoring", AIAA/SAE paper 72-1082, November
- Urban, L. A., 1974: "Parameter selection for multiple fault diagnostics of gas turbine engines", 74-GT-62, Journal of Engineering for Gas Turbines and Power, Transactions of the ASME
- Urban, L.A., and Volponi, A.J., 1992: "Mathematical methods of relative engine performance diagnostics", SAE paper 922048
- Volponi, A.J., 1994: "Sensor error compensation in engine performance diagnostics", paper ASME 94-GT-58
- Vivian, B.M. , and Singh, R, 1995: "Application of Expert System technology to the gas path analysis of a gas turbine turboprop", EPF 95-01, AIDAA/AAAF/DGLR/RAeS 5th European Forum, Pisa, Italy
- Walsh, P.P., and Fletcher, P., 1998: "Gas turbine performance", Blackwell Science
- Weidong, H., Kechang, W. and Qizhi, C., 1996: "Sensor failure detection and data recovery based on neural networks", AIAA 96-2932, 32nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference, July 1-3, Lake Buena Vista, Florida

- Whitehead, B., Ferber, H., and Ali, M., 1990: "Neural network approach to SSME health monitoring", AIAA 90-2259, AIAA/SAE/ASME/ASEE 26th Joint Propulsion Conference, July 16-18, Orlando, Florida
- Whitehead, B., Kiech, E., and Ali, M., 1990: "Rocket engine diagnostics using Neural Networks", AIAA 90-1892, AIAA/SAE/ASME/ASEE 26th Joint Propulsion Conference, July 16-18, Orlando, Florida
- Willan, U., 1990: "Integration von modellbezogener und wissenbasierter Diagnose am Beispiel eines Turboflugtriebwerkes", PhD dissertation, German Armed Forces University, Hamburg
- Zedda, M., 1996: "Diagnostica di una turbina a gas con reti neurali", thesis of Aeronautical Engineering, University of Pisa, Italy
- Zedda, M. and Singh, R., 1998: "Fault diagnosis of a turbofan engine using neural networks: a quantitative approach", AIAA 98-3062, AIAA/SAE/ASME/ASEE 34th Joint Propulsion Conference and Exhibit, July 13-15, Cleveland, OH
- Zedda, M. and Singh, R., 1999a: "Gas turbine engine and sensor fault diagnosis using optimisation techniques", paper AIAA 99-2530, AIAA/SAE/ASME/ASEE 35th Joint Propulsion Conference and Exhibit, June 20-24, Los Angeles, CA
- Zedda, M. and Singh, R., 1999b: "Neural network based sensor validation for gas turbines", IMechE Seminar on Gas Turbine Life: Diagnostics and Prognostics, July 6, Derby, U.K.
- Zedda, M. and Singh, R., 1999c: "Gas turbine engine and sensor diagnostics", IS10/UNK010, XIV International Symposium on Air-Breathing Engines (ISABE), 5-10 September, Florence, Italy
- Zedda, M., Gulati, A. and Singh, R., 1999: "Advanced diagnostic techniques for gas turbines: an overview", 5th Industrial & Power Gas Turbine Operations and Maintenance, London, U.K., 26-28th October

APPENDIX A

Detection, Isolation and Accommodation algorithm for Kalman filtering

PW have developed an algorithm, which can be used along with the KF, to detect, isolate and accommodate single measurement biases.

If the Kalman filter is used to estimate both engine and sensor faults (see eq. (2.33)) and the Kalman gain matrix (see eq. (2.20)) is partitioned as follows:

$$K_k = [K_{ek} \vdots K_{sk}] \tag{a.1}$$

to separate performance parameters from measurement biases, the state estimate update equation (see eq. 2.18) becomes:

$$\hat{\mathbf{X}} = \overline{\mathbf{X}} + K \cdot [\mathbf{z} - H \cdot \overline{\mathbf{X}}] \tag{a.2}$$

where a single stage estimation is assumed and therefore the index k is not used. The caret "^" is used for the estimate, the overline "¯" for the a priori value. If a time series has to be analysed, the estimate at the previous stage becomes the a priori value for the current stage and the technique shown below still applies.
It is reminded that the state vector $\mathbf{X}$ contains both the performance parameters $\mathbf{x}$ and the measurement biases $\mathbf{b}$ according to (2.33). Equivalently, eq. (a.2) can be split in two vector equations, one for $\mathbf{x}$ and one for $\mathbf{b}$:

$$\hat{\mathbf{x}} = \overline{\mathbf{x}} + K_e \cdot [\mathbf{z} - H_e \cdot \overline{\mathbf{x}} - H_s \cdot \overline{\mathbf{b}}] \tag{a.3}$$

$$\hat{\mathbf{b}} = \overline{\mathbf{b}} + K_s \cdot [\mathbf{z} - H_e \cdot \overline{\mathbf{x}} - H_s \cdot \overline{\mathbf{b}}] \tag{a.4}$$

Sensor fault isolation is accomplished by checking on the values of the following quantity:

$$\frac{\left|\hat{b}_k - \overline{b}_k\right|}{T_k} \qquad k = 1,\ldots,M_{bias} \tag{a.5}$$

where:

- M_{bias} is the number of sensors which may be affected by faults
- T_k is the threshold for the k-the possibly biased measurement.

The way thresholds can be set is explained later on.

It may be worthwhile to stress that the method accounts for single measurement biases and that only M_{bias} out of M sensor are allowed to be affected by faults. As a matter of fact, there are measurements such as spool speeds that can be safely regarded as fault-free. The reason for this assumption is twofold:

- the sensor themselves can be considered somewhat more reliable than other sensors (e.g. thermocouples)
- occurrence of a sensor fault is likely to manifests itself clearly and not by means of slow drifts.

The effect of biases can be separated by introducing the following quantity:

$$\mathbf{y} = \mathbf{z} - H_e \cdot \bar{\mathbf{x}} \tag{a.5}$$

so that the components of $\hat{\mathbf{x}}$ and $\hat{\mathbf{b}}$ can be written as follows:

$$\hat{x}_i = \bar{x}_i + \sum_{j=1}^{M} K_{eij} \cdot \left[y_j - \sum_{n=1}^{M_{bias}} H_{sjn} \cdot \bar{b}_n \right] \qquad i = 1,\ldots,N \tag{a.6}$$

$$\hat{b}_i = \bar{b}_i + \sum_{j=1}^{M} K_{sij} \cdot \left[y_j - \sum_{n=1}^{M_{bias}} H_{sjn} \cdot \bar{b}_n \right] \qquad i = 1,\ldots,M_{bias} \tag{a.7}$$

respectively.
An index set Ω_k is introduced, which selects those rows from the k-th column of the matrix H_s which are non-zero:

$$\Omega_k = \{n | H_{snk} \neq 0\} \tag{a.8}$$

In terms of this new index set, eq. (a.6) can be re-written as follows:

$$\hat{x}_i = \bar{x}_i + \sum_{n \in \Omega_k} K_{ein} \cdot \left[y_n - \sum_{j=1}^{M} H_{snj} \cdot \bar{b}_j \right] + \varphi_i \tag{a.9}$$

where:

$$\varphi_i = \sum_{n \notin \Omega_k} K_{ein} \cdot \left[y_n - \sum_{j=1}^{M} H_{snj} \cdot \bar{b}_j \right] \tag{a.10}$$

It should be noted that φ_i is independent of $\bar{b}_k$.
In order to isolate the effect of the k-th bias $\bar{b}_k$, eq. (a.9) is again modified as follows:

$$\hat{x}_i = \bar{x}_i + a_i \cdot \bar{b}_k + c_i \tag{a.11}$$

where:

$$a_i = -\sum_{n\in\Omega_k} K_{ein}\cdot H_{snk} \tag{a.12}$$

$$c_i = \sum_{n\in\Omega_k} K_{ein}\cdot\left[y_n - \sum_{\substack{j=1\\ j\neq k}}^{M} H_{snj}\cdot\bar{b}_j\right] + \varphi_i \tag{a.13}$$

It should be noted that also a_i and c_i are independent of $\bar{b}_k$.

A similar procedure can be developed for $\hat{b}_i$ as well:

$$\hat{b}_i = \bar{b}_i + \alpha_i\cdot\bar{b}_k + \gamma_i \tag{a.14}$$

where:

$$\alpha_i = -\sum_{n\in\Omega_k} K_{sin}\cdot H_{snk} \tag{a.15}$$

$$\gamma_i = \sum_{n\in\Omega_k} K_{sin}\cdot\left[y_n - \sum_{\substack{j=1\\ j\neq k}}^{M} H_{snj}\cdot\bar{b}_j\right] + \psi_i \tag{a.16}$$

$$\psi_i = \sum_{n\in\Omega_k} K_{sin}\cdot\left[y_n - \sum_{j=1}^{M} H_{sni}\cdot\bar{b}_j\right] \tag{a.17}$$

Once $\hat{x}_i$ and $\hat{b}_i$ are expressed in (a.9) and (a.14) as quantities linearly and explicitly dependent on $\bar{b}_k$, the estimate $\hat{b}_k$ of the bias can be obtained by minimising the following objective function:

$$J(\bar{b}_k) = \sum_{j=1}^{M}(\hat{b}_j - \bar{b}_j)^2 + \sum_{j=1}^{N}(\hat{x}_j - \bar{x}_j)^2 \tag{a.18}$$

with respect to $\bar{b}_k$. Expanding (a.18) and equating to zero the first derivative with respect to $\bar{b}_k$, the desired estimate is obtained:

$$\hat{b}_j = \bar{b}_j + \frac{[KH_s]_j\cdot(\hat{\mathbf{x}} - \bar{\mathbf{x}})}{\left|[KH_s]_j\right|^2} \tag{a.19}$$

which is the same as (2.41). The rationale behind the use of the objective function (a.18) is that discrepancies between expected and a priori deviations of performance parameters and measurement biases are minimised by adjusting $\bar{b}_k$.

Choice of the thresholds T_k is obviously crucial: a coarse setting of these parameters can completely impair the effectiveness of the recovery technique. An upper and lower bound of the threshold levels, though, can be obtained by imposing two conditions.
Let a bias θ_k affect the k-th measurement. The quantity due to the bias in measurement k that affects the i-the bias estimation is given by:

$$[K_s \cdot H_s]_{ik} \cdot \theta_k \tag{a.20}$$

If this definition is given:

$$K_s \cdot H_s = D \tag{a.21}$$

the following inequality has to be satisfied to prevent the algorithm from isolating a bias in the i -th rather than in the k-th measurement:

$$\frac{D_{ik} \cdot \theta_k}{T_i} < \frac{D_{kk} \cdot \theta_k}{T_k} \qquad \forall i \neq k \tag{a.22}$$

If this definition is introduced for the lower bound:

$$L_{ik} = \frac{D_{ik}}{D_{kkl}} \qquad \forall i \neq k \tag{a.23}$$

(a.22) reduces to:

$$\frac{T_i}{T_k} > L_{ik} \tag{a.24}$$

Similarly, a bias in the i -th measurement should not be perceived as a bias in the k-the measurement and thus:

$$\frac{D_{ki} \cdot \theta_i}{T_k} < \frac{D_{ii} \cdot \theta_i}{T_i} \tag{a.25}$$

If the upper bound is defined as follows:

$$U_{ik} = \frac{D_{ii}}{D_{ki}} \tag{a.26}$$

the inequality (a.25) becomes:

$$\frac{T_i}{T_k} < U_{ik} \tag{a.27}$$

Combining (a.24) and (a.27) produces:

$$L_{ik} \cdot T_k < T_i < U_{ik} \cdot T_k \tag{a.28}$$

In this way the choice of the threshold can be checked in terms of upper and lower bounds to reduce the probability of misdiagnosis of a bias location.

APPENDIX B

Optimisation technique for the Influence Coefficient Matrix

The matrix Q used in eq. (2.55) is dependent on measurements taken up- and downstream of the component considered and on its characteristic maps (see eq. (2.54)). As the characteristics and especially their gradients are assumed to be known only approximately, an optimisation technique has been developed (Lunderstaedt and Fiedler, 1988) to modify the matrix properly.
Any element of the matrix Q can be written as:

$$q_{ij} = a_{ij} + b_{ij}\rho_i + c_{ij}\varepsilon_i \tag{b.1}$$

where ρ_i and ε_i are the gradients of the considered performance parameter with respect to two quantities, which are chosen in order to avoid the presence of very large values of the derivatives. In that case large errors are likely to occur.
Whereas a_{ij}, b_{ij} and c_{ij} are known precisely, ρ_i and ε_i are actually known only approximately.
The following definitions are given:

$$a_i = \sum_{j=1}^{M} a_{ij}\Delta y_j \tag{b.2}$$

$$b_i = \sum_{j=1}^{M} b_{ij}\Delta y_j \tag{b.3}$$

$$c_i = \sum_{j=1}^{M} c_{ij}\Delta y_j \tag{b.4}$$

where M is the number of measurements.
The generic performance parameter can be written:

$$\Delta x_i = a_i + b_i\rho_i + c_i\varepsilon_i \qquad i = 1,...,N \tag{b.5}$$

where N is the number of performance parameters to estimate.
If fault free measurements are assumed, the case without model errors is described by:

$$\Delta x_{i0} = a_i + b_i\rho_{i0} + c_i\varepsilon_{i0} \tag{b.6}$$

The model errors are expressed as follows:

$$\rho_{i0} = \rho_i + \Delta\rho_i \tag{b.7}$$

$$\varepsilon_{i0} = \varepsilon_i + \Delta\varepsilon_i \tag{b.8}$$

and similarly:

$$\delta\Delta x_i = \Delta x_i - \Delta x_{i0} \tag{b.9}$$

Eq. (b.5) then becomes:

$$\delta\Delta x_i = b_i \Delta\rho_i + c_i \Delta\varepsilon_i \tag{b.10}$$

K different engine model errors are simulated in the full non-linear model and the following system has to be solved for every performance parameter:

$$\delta\Delta\mathbf{x}_i = \mathbf{Y}_i \cdot \begin{bmatrix} \Delta\rho_i \\ \\ \Delta\varepsilon_i \end{bmatrix} \tag{b.11}$$

where:

- $\delta\Delta\mathbf{x}_i \in R^K$ is the model error vector
- $\mathbf{Y}_i \in R^{Kx2}$ is the "measuring" matrix.

The model errors are obtained by means of the pseudo-inverse matrix:

$$\begin{bmatrix} \Delta\rho_i \\ \\ \Delta\varepsilon_i \end{bmatrix} = (\mathbf{Y}_i^T \mathbf{Y}_i)^{-1} \mathbf{Y}_i^T \delta\Delta\mathbf{x}_i \tag{b.12}$$

The so-determined model errors are added to the initial value of the gradients of the characteristics and a new Q is calculated.

Two remarks have to be done about this technique:

- the optimisation outlined above is suitable for performance parameters of engine components whose behaviour is represented by characteristic maps. A different optimisation technique has to be devised and used for parameters not related to a characteristic map
- the corrections $\Delta\rho_i$ and $\Delta\varepsilon_i$ calculated with eq. (b.12) are often so large that the new estimated value of the gradient is meaningless. Therefore the gradient errors must be constrained and this can be done by means of a penalty function technique. Thus the estimation procedure becomes non-linear and requires an iterative solution.

APPENDIX C

Single stage Bayesian estimator

The inverse functions are respectively:

$$\mathbf{F}_1^{-1}(\mathbf{Z}) = \mathbf{h}^{-1}(\mathbf{z} - \mathbf{w}) \tag{c.1}$$

$$\mathbf{F}_2^{-1}(\mathbf{Z}) = \mathbf{w} \tag{c.2}$$

It is assumed that the number of performance parameters is equal to the number of measurements, so that the inversion of the function in (c.1) is straightforward.
Since all performance parameters are assumed to be independent of each other:

$$p_{\mathbf{X}}(\mathbf{F}^{-1}(\mathbf{Z})) = p_{x_1}(h_1^{-1}(\mathbf{z} - \mathbf{w})) \cdot p_{x_2}(h_2^{-1}(\mathbf{z} - \mathbf{w})) \cdot \ldots\ldots\ldots\ldots \cdot p_{x_m}(h_m^{-1}(\mathbf{z} - \mathbf{w})) \tag{c.3}$$

According to (c.1) and (c.2), the Jacobian matrix of eq. (2.125) becomes:

$$\frac{\partial \mathbf{F}^{-1}(\mathbf{Z})}{\partial \mathbf{Z}} = \begin{bmatrix} \dfrac{\partial \mathbf{F}_1^{-1}(\mathbf{Z})}{\partial \mathbf{Z}} & \dfrac{\partial \mathbf{F}_2^{-1}(\mathbf{Z})}{\partial \mathbf{w}} \\ 0 & I \end{bmatrix} \tag{c.4}$$

The elements of the Jacobian matrix can be calculated as follows:

$$\frac{\partial F_i^{-1}}{\partial z_j} = \frac{\partial}{\partial z_j}(h_i^{-1}(\mathbf{z} - \mathbf{w})) = \frac{\partial h_i^{-1}(\mathbf{z})}{\partial z_j} \tag{c.5}$$

if $i \le m, j \le m$, where m is the number of measurements, and

$$\frac{\partial F_i^{-1}(\mathbf{Z})}{\partial w_j} = -\frac{\partial h_i^{-1}(\mathbf{v})}{v_j} \tag{c.6}$$

if $i \le m, j > m$.
Eventually the desired probability density function is calculated through integration:

$$p_{\mathbf{z}}(\mathbf{z}) = \int\int \ldots \int p_{\mathbf{Z}}(\mathbf{Z}) dw_1 dw_2 \ldots\ldots dw_m \tag{c.7}$$

APPENDIX D

Approximated Bayesian estimator

A set of input (control) vectors is given:

$$\mathbf{Z}_N = [\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N]^T \tag{d.1}$$

The set of measured quantities is:

$$\mathbf{X}_N = [\mathbf{x}_1^o, \mathbf{x}_2^o, ..., \mathbf{x}_N^o]^T \tag{d.2}$$

Time is discretised: $t_n = n \cdot \Delta t$, the "o" means measured or "observed" and the caret indicates the measured quantities. In the original treatment measurement noise is neglected (Katafygiotis, 1991), because the accelerometers used for the study of structural dynamics are accurate enough. In the case of gas turbine engines, though, such assumption would be unacceptable.
If $\mathbf{a}$ is the performance parameter vector, containing the uncertain parameters of the engine model M, its joint probability density function is $\pi_{\mathbf{a}}(\mathbf{a})$. The ***model output*** is:

$$\mathbf{q}(n;\mathbf{a}) = \mathbf{q}(n;\mathbf{a},\mathbf{Z}_N,M) \tag{d.3}$$

The ***engine output*** is $\mathbf{x}(n)$. Both model and engine outputs contain measured and unmeasured quantities. Therefore the vectors are arranged so as to have the measured quantities in the first N_o elements:

$$\mathbf{x}(n) = [\mathbf{x}^o(n), \mathbf{x}^u(n)]^T \tag{d.4}$$

$$\mathbf{q}(n) = [\mathbf{q}^o(n), \mathbf{q}^u(n)]^T \tag{d.5}$$

where "u" means unmeasured.
As stated above, the model uncertainty is expressed through a set of parameters defining the spread of the performance parameters, i.e. their standard deviation. The full vector of the uncertainties is:

$$\tilde{\mathbf{a}} = [\mathbf{a}, \boldsymbol{\sigma}]^T \tag{d.6}$$

where $\boldsymbol{\sigma}$ collects the standard deviations.
In this way the class of engine models M is expanded to the class of probabilistic model Mp, parameterised in terms of $\tilde{\mathbf{a}}$. Given the model, the uncertain parameters and the control history, the output pdf may be written in different ways:

$$p(\mathbf{X}_N \mid \tilde{\mathbf{a}}, \mathbf{Z}_N, Mp) = g_N(\mathbf{X}_N; \tilde{\mathbf{a}}, \mathbf{Z}_N) = g_N'(\mathbf{X}_N^o, \mathbf{X}_N^u; \tilde{\mathbf{a}}, \mathbf{Z}_N) \tag{d.7}$$

where :

$$\mathbf{X}_N^o = [\mathbf{x}_1^o, \mathbf{x}_2^o, ..., \mathbf{x}_N^o]^T \tag{d.8}$$

$$\mathbf{X}_N^u = [\mathbf{x}_1^u, \mathbf{x}_2^u, ..., \mathbf{x}_N^u]^T \tag{d.9}$$

Similarly, the prior pdf of the uncertain parameter vector can be written:

$$\pi_{\tilde{\mathbf{a}}}(\tilde{\mathbf{a}}) = p(\tilde{\mathbf{a}} \mid Mp) \tag{d.10}$$

The following treatment is based on the *model error method* (Katafygiotis, 1991). The output error is defined as the difference between model and engine output:

$$\mathbf{e}(n) = \mathbf{x}(n) - \mathbf{q}(n; \tilde{\mathbf{a}}) \tag{d.11}$$

The joint pdf of the output error, given the probabilistic model, is defined:

$$p(\mathbf{e}(1), \mathbf{e}(2), ..., \mathbf{e}(N) \mid \boldsymbol{\sigma}, P) = h_N(\mathbf{e}(1), \mathbf{e}(2), ..., \mathbf{e}(N)) \tag{d.12}$$

Eq. (d.7) can be written:

$$g_N(\mathbf{X}_N; \tilde{\mathbf{a}}, \mathbf{Z}_N) = h_N((\mathbf{x}(1) - \mathbf{q}(1; \mathbf{a})), ..., (\mathbf{x}(N) - \mathbf{q}(N; \mathbf{a}); \boldsymbol{\sigma}) \tag{d.13}$$

The following vector is defined:

$$\mathbf{E}_N = [\mathbf{e}(1), ..., \mathbf{e}(N)] \tag{d.14}$$

If the Bayes' theorem is used repeatedly:

$$\begin{aligned} p(\mathbf{E}_N \mid \boldsymbol{\sigma}, P) &= p(\mathbf{e}(N) \mid \mathbf{E}_{N-1}, \boldsymbol{\sigma}, P) \cdot p(\mathbf{E}_{N-1} \mid \boldsymbol{\sigma}, P) = \\ &= ..... = \prod_{n=1}^{N} p(\mathbf{e}(n) \mid \mathbf{E}_{N-1}, \boldsymbol{\sigma}, P) \end{aligned} \tag{d.15}$$

Defining:

$$h_n'(\mathbf{e}(n); \mathbf{E}_{n-1}, \boldsymbol{\sigma}) = p(\mathbf{e}(n) \mid \mathbf{e}(1), ..., \mathbf{e}(n-1), \boldsymbol{\sigma}, P) \tag{d.16}$$

leads to:

$$h_N = \prod_{n=1}^{N} h_n' \tag{d.17}$$

The process $\mathbf{e}(n)$ is assumed to be mean zero, Gaussian, white noise. Since the noise is white:

$$h_n'(\mathbf{e}(n);\mathbf{E}_{n-1},\boldsymbol{\sigma}) = h'(\mathbf{e}(n);\boldsymbol{\sigma}) = G(\mathbf{0},\Sigma(\boldsymbol{\sigma})) \tag{d.18}$$

where $G(\mathbf{0},\Sigma(\boldsymbol{\sigma}))$ is a multidimensional Gaussian distribution with zero mean values and constant covariance matrix $\Sigma(\boldsymbol{\sigma})$.
Eq. (d.17) then becomes:

$$h_N(\mathbf{E}_N;\boldsymbol{\sigma}) = \frac{1}{(2\pi)^{NN_R/2}\left|\Sigma(\boldsymbol{\sigma})\right|^{N/2}} \cdot e^{-\frac{1}{2}\sum_{n=1}^{N}[\mathbf{e}^T(n)\cdot\Sigma^{-1}(\boldsymbol{\sigma})\cdot\mathbf{e}(n)]} \tag{d.19}$$

where N_R is the number of output quantities.
Using (d.12) for (d.19) gives:

$$g_N(\mathbf{X}_N;\tilde{\mathbf{a}},\mathbf{Z}_N) = \frac{1}{(2\pi)^{NN_r/2}\cdot\left|\Sigma(\boldsymbol{\sigma})\right|^{N/2}} \cdot e^{-\frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}(n)-\mathbf{q}(n;\mathbf{a}))^T\cdot\Sigma^{-1}(\boldsymbol{\sigma})\cdot(\mathbf{x}(n)-\mathbf{q}(n;\mathbf{a}))} \tag{d.20}$$

The covariance matrix $\Sigma(\boldsymbol{\sigma})$ is usually diagonal. Since gas turbines' instrumentation set is made of sensors usually characterised by different noise levels, every measurement has its own standard deviation in principle. The covariance matrix is arranged like this:

$$\Sigma(\boldsymbol{\sigma}) = \begin{bmatrix} \Sigma^o(\boldsymbol{\sigma}^o) & 0 \\ 0 & \Sigma^u(\boldsymbol{\sigma}^u) \end{bmatrix} \tag{d.21}$$

where:

- the standard deviation vector is re-arranged as follows:

$$\boldsymbol{\sigma} = [\boldsymbol{\sigma}^o,\boldsymbol{\sigma}^u]^T \tag{d.22}$$

- the uncertain parameters are arranged this way:

$$\mathbf{a} = [\mathbf{a}^i,\mathbf{a}^{ni}]^T \tag{d.23}$$

where $\mathbf{a}^i$ contains those uncertain parameters affecting at least one measured output, $\mathbf{a}^{ni}$ contains the remainder.

It should be noted that the unmeasured output quantities are dependent on the full vector $\mathbf{a}$. In formulae:

$$q_i^o = q_i^o(n;\mathbf{a}^i) \tag{d.24}$$

$$q_i^u = q_i^u(n;\mathbf{a}) \tag{d.25}$$

The following definitions are useful:

$$\mathbf{e}^o(n;\mathbf{a}^i) = \mathbf{x}^o(n) - \mathbf{q}^o(n;\mathbf{a}^i) \tag{d.26}$$

$$\mathbf{e}^u(n;\mathbf{a}) = \mathbf{x}^u(n) - \mathbf{q}^u(n;\mathbf{a}) \tag{d.27}$$

Integrating (d.20) wrt $d\mathbf{X}^u$ and $d\mathbf{X}^o$ respectively provides:

$$p(\mathbf{X}_N^o \mid \tilde{\mathbf{a}}, \mathbf{Z}_N, Mp) = \frac{1}{(2\pi)^{NN_O/2}\left|\Sigma^o(\boldsymbol{\sigma}^o)\right|^{N/2}} e^{\left(-\frac{1}{2}\sum_{n=1}^{N}\sum_{i=1}^{N_O}\frac{e_i^o(n;\mathbf{a}^i)^2}{\left|\Sigma^o(\boldsymbol{\sigma}^o)\right|_{ii}}\right)} = \tag{d.28}$$

$$= f_N^O(\mathbf{X}_N^O; \mathbf{a}^i, \boldsymbol{\sigma}^o, \mathbf{Z}_N)$$

$$p(\mathbf{X}_N^u \mid \tilde{\mathbf{a}}, \mathbf{Z}_N, Mp) = \frac{1}{(2\pi)^{NN_u/2}\left|\Sigma^u(\boldsymbol{\sigma}^u)\right|^{N/2}} e^{\left(-\frac{1}{2}\sum_{n=1}^{N}\sum_{i=1}^{N_u}\frac{e_i^o(n;\mathbf{a})^2}{\left|\Sigma^u(\boldsymbol{\sigma})\right|_{ii}}\right)} = \tag{d.29}$$

$$= f_N^u(\mathbf{X}_N^u; \tilde{\mathbf{a}}, \mathbf{Z}_N)$$

Bayes' theorem states:

$$p(\tilde{\mathbf{a}} \mid \mathbf{Z}_N, \mathbf{X}_N, Mp) = \frac{p(\hat{\mathbf{X}}_N \mid \tilde{\mathbf{a}}, \hat{\mathbf{Z}}_N, Mp) \cdot p(\tilde{\mathbf{a}} \mid Mp)}{p(\hat{\mathbf{X}}_N \mid \hat{\mathbf{Z}}_N, Mp)} = K \cdot f_N^o(\hat{\mathbf{X}}_N; \mathbf{a}_i, \boldsymbol{\sigma}^o) \cdot \pi_{\tilde{\mathbf{a}}}(\tilde{\mathbf{a}}) \tag{d.30}$$

where:

$$K^{-1} = p(\hat{\mathbf{X}}_N \mid \hat{\mathbf{Z}}_N, Mp) = \int_{S(\tilde{\mathbf{a}})} p(\hat{\mathbf{X}}_N \mid \tilde{\mathbf{a}}, \hat{\mathbf{Z}}_N, Mp) \cdot p(\tilde{\mathbf{a}} \mid Mp) d\tilde{\mathbf{a}} \tag{d.31}$$

with $S(\tilde{\mathbf{a}})$ as the domain of $\tilde{\mathbf{a}}$.
Since:

$$\pi_{\tilde{\mathbf{a}}}(\tilde{\mathbf{a}}) = p(\tilde{\mathbf{a}} \mid Mp) = p(\mathbf{a}^{ni}, \boldsymbol{\sigma}^{ni} \mid \mathbf{a}^{i}, \boldsymbol{\sigma}^{o}, Mp) \cdot p(\mathbf{a}^{i}, \boldsymbol{\sigma}^{o} \mid Mp) \tag{d.32}$$

where

$$p(\mathbf{a}^{i}, \boldsymbol{\sigma}^{o} \mid Mp) = \int_{S(\mathbf{a}^{ni}, \boldsymbol{\sigma}^{ni})} p(\tilde{\mathbf{a}} \mid Mp) d\mathbf{a}^{ni} d\boldsymbol{\sigma}^{ni} \equiv \pi_{\mathbf{a}^{i}, \boldsymbol{\sigma}^{O}}(\mathbf{a}^{i}, \boldsymbol{\sigma}^{O}) \tag{d.33}$$

substituting (d.32) in (d.30) and integrating wrt $\mathbf{a}^{ni}, \boldsymbol{\sigma}^{ni}$:

$$p(\mathbf{a}_i, \boldsymbol{\sigma}^{o} \mid \hat{\mathbf{Z}}_N, \hat{\mathbf{X}}_N, Mp) = K \cdot f_N^{o}(\hat{\mathbf{X}}_N; \mathbf{a}^{i}, \boldsymbol{\sigma}^{o}, \hat{\mathbf{Z}}_N) \cdot \pi_{\mathbf{a}^{i}, \boldsymbol{\sigma}^{O}}(\mathbf{a}^{i}, \boldsymbol{\sigma}^{O}) \tag{d.34}$$

Aim of the estimation is the maximisation of $p(\mathbf{a}_i \boldsymbol{\sigma}^{o} \mid \hat{\mathbf{Z}}_N, \hat{\mathbf{X}}_N, Mp)$ wrt $[\mathbf{a}^{i}, \boldsymbol{\sigma}^{o}]$ to find a maximum likelihood solution. If the pdf $\pi_{\mathbf{a}^{i}, \boldsymbol{\sigma}^{O}}(\mathbf{a}^{i}, \boldsymbol{\sigma}^{O})$ is slowly varying, the optimal solution is given by those vectors $[\hat{\mathbf{a}}^{i}, \hat{\boldsymbol{\sigma}}^{o}]$ which maximise $f_N^{o}(\hat{\mathbf{X}}_N; \mathbf{a}^{i}, \boldsymbol{\sigma}^{o}, \hat{\mathbf{Z}}_N)$. It can be shown (Katafygiotis, 1991) that the analytical requirement for the prior pdf $\pi_{\mathbf{a}^{i}, \boldsymbol{\sigma}^{O}}(\mathbf{a}^{i}, \boldsymbol{\sigma}^{O})$ is that it has to be constant in a domain whose radius is $O(N^{-1})$. In practice, for a slowly varying pdf and a large number of measurements, this requirement is always satisfied and the maximisation task is simplified.
In the original application to gas turbine diagnostics (Consumi, 1996; Consumi and D'Agostino, 1997) all measurements were supposed to be affected by the same level of noise:

$$\sigma_i^{o} = \sigma_o^{o} \qquad \text{for every } i \tag{d.35}$$

In this case, the objective function to minimise simply results:

$$J(\mathbf{a}^{i}) = \sum_{n=1}^{N} [\mathbf{x}^{o}(t_n) - \mathbf{q}^{o}(t_n, \mathbf{a}^{i})]^2 \tag{d.36}$$

However, in general gas turbine sensors show different noise levels. A modification to the technique is here proposed, which is able to account for this real-world effect. The noise affecting the various measurements is usually uncorrelated and therefore the matrix $\Sigma^{o}(\boldsymbol{\sigma}^{o})$ is diagonal.
Taking the logarithm of (d.34):

$$\ln(p(\mathbf{a}_i,\boldsymbol{\sigma}^o \mid \hat{\mathbf{Z}}_N,\hat{\mathbf{X}}_N,Mp)) = \ln(K) - \frac{NN_o}{2}\cdot\ln(2\pi) + \ln(\cdot\pi_{\mathbf{a}^i,\boldsymbol{\sigma}^O}(\mathbf{a}^i,\boldsymbol{\sigma}^O)) +$$
$$-\frac{N}{2}\cdot\ln(\left|\Sigma^O(\boldsymbol{\sigma}^O)\right|) - \frac{1}{2}\cdot\sum_{n=1}^{N}\mathbf{e}(n)^T\cdot\Sigma^O(\boldsymbol{\sigma}^O)^{-1}\cdot\mathbf{e}(n) \qquad \text{(d.37)}$$

The following definitions are given:

$$C = \ln(K) - \frac{NN_o}{2}\cdot\ln(2\pi) \qquad \text{(d.38)}$$

$$D = \ln(\cdot\pi_{\mathbf{a}^i,\boldsymbol{\sigma}^O}(\mathbf{a}^i,\boldsymbol{\sigma}^O)) \qquad \text{(d.39)}$$

C is constant, D is assumed to be constant in a domain whose radius is $O(N^{-1})$ (see above).
Since $\Sigma^o(\boldsymbol{\sigma}^o)$ is diagonal:

$$\ln(p(\mathbf{a}_i,\boldsymbol{\sigma}^o \mid \hat{\mathbf{Z}}_N,\hat{\mathbf{X}}_N,Mp)) = C + D - N\cdot\sum_{i=1}^{N_o}\ln(\sigma_i^o) - \frac{1}{2}\cdot\sum_{n=1}^{N}\sum_{i=1}^{N_o}\left[\frac{\mathbf{x}_i^o(n)-\mathbf{q}_i^o(n;\mathbf{a}^i)}{\sigma_i^o}\right]^2 \qquad \text{(d.40)}$$

Extrema wrt any σ_j^o can be found by imposing:

$$\frac{\partial\ln(p(\mathbf{a}_i,\boldsymbol{\sigma}^o \mid \hat{\mathbf{Z}}_N,\hat{\mathbf{X}}_N,Mp))}{\partial\sigma_j^o} = 0 \qquad \text{(d.41)}$$

Solving eq. (d.41) produces:

$$\hat{\sigma}_j^{o2} = \frac{1}{N}\cdot\sum_{n=1}^{N}\left[\mathbf{x}_i^o(n)-\mathbf{q}_i^o(n;\mathbf{a}^i)\right]^2 \qquad \text{(d.42)}$$

Calculating all σ_j^o and substituting for (d.40):

$$\ln(p(\mathbf{a}_i,\hat{\boldsymbol{\sigma}}^o \mid \hat{\mathbf{Z}}_N,\hat{\mathbf{X}}_N,Mp)) = C + D - \frac{N}{2}\cdot\ln(\prod_{i=1}^{N_o}\hat{\sigma}_i^{o2}) - \frac{NN_o}{2} \qquad \text{(d.43)}$$

The maximum likelihood solution can the be obtained by calculating the minimum of:

$$J(\mathbf{a}^i) = \frac{1}{N}\cdot\prod_{i=1}^{N_o}\sum_{n=1}^{N}\left[\mathbf{x}_i^o(n)-\mathbf{q}_i^o(n;\mathbf{a}^i)\right]^2 \qquad \text{(d.44)}$$

In conclusion, accounting for the actually different measurement non-repeatabilities changes (d.36) to (d.44).

APPENDIX E

Extended Kalman Filter (EKF)

The equations required by the continuous-discrete EKF are as usual:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{w}(t) \qquad \text{dynamics model} \qquad \text{(e.1)}$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}(t_k)) + \mathbf{v}_k \qquad k = 1,2..... \quad \text{measurement model} \qquad \text{(e.2)}$$

where:

- $\mathbf{w}(t)$ is the continuous process noise, assumed to be white, Gaussian, zero mean, with covariance matrix $Q(t)$:

$$\mathbf{w}(t) \sim N(0, Q(t)) \qquad \text{(e.3)}$$

- $\mathbf{v}_k$ is the discrete measurement noise, assumed to be white, Gaussian, zero mean, with covariance matrix R_k:

$$\mathbf{v}_k \sim N(0, R_k) \qquad \text{(e.4)}$$

Initial conditions are assumed for the state vector and the covariance matrix:

$$\mathbf{x}(0) \sim N(\mathbf{x}(0), P_0) \qquad \text{(e.5)}$$

Measurement and process noise are uncorrelated:

$$E[\mathbf{w}(t) \cdot \mathbf{v}_k] = 0 \qquad \text{for all } k \text{ and } t \qquad \text{(e.6)}$$

The following definitions are given:

$$F(\hat{\mathbf{x}}(t), t) = \left. \frac{\partial \mathbf{f}(\hat{\mathbf{x}}(t), t)}{\partial \mathbf{x}(t)} \right|_{\mathbf{x}(t)=\hat{\mathbf{x}}(t)} \qquad \text{(e.7)}$$

$$H_k(\hat{\mathbf{x}}(-)) = \left. \frac{\partial \mathbf{h}_k(\mathbf{x}(t_k))}{\partial \mathbf{x}(t_k)} \right|_{\mathbf{x}(t_k)=\hat{\mathbf{x}}(-)} \qquad \text{(e.8)}$$

The following equations make up the EKF:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), t) \qquad \text{state estimate propagation} \qquad \text{(e.9)}$$

$$\dot{P}(t) = F(\hat{\mathbf{x}}(t),t)P(t) + P(t)F^T(\hat{\mathbf{x}}(t),t) + Q(t) \quad \text{error covariance propagation} \quad \text{(e.10)}$$

$$\hat{\mathbf{x}}_k(+) = \hat{\mathbf{x}}_k(-) + K_k[\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_k(-))] \quad \text{state estimate update} \quad \text{(e.11)}$$

$$P_k(+) = [I - K_k H_k(\hat{\mathbf{x}}_k(-))]P_k(-) \quad \text{error covariance update} \quad \text{(e.12)}$$

$$K_k = P_k(-)H_k^T(\hat{\mathbf{x}}_k(-))[H_k(\hat{\mathbf{x}}_k(-))P_k(-)H_k^T(\hat{\mathbf{x}}_k(-)) + R_k]^{-1} \quad \text{gain matrix} \quad \text{(e.13)}$$

Iterated Extended Kalman Filter (IEKF)

The estimation provided by the EKF can be improved by repeatedly calculating $\hat{\mathbf{x}}_k(+)$, K_k and $P_k(+)$, each time linearising about the most recent estimate.
The non-linear function $\mathbf{h}$ is expanded in Taylor's series and only the first order term is used:

$$\mathbf{h}_k(\mathbf{x}_{k,i}) \cong \mathbf{h}_k(\hat{\mathbf{x}}_{k,i}(+)) + H_k(\hat{\mathbf{x}}_{k,i}(+))(\mathbf{x}_k - \hat{\mathbf{x}}_{k,i}(+)) \quad \text{(e.14)}$$

where:

$$H_k(\hat{\mathbf{x}}_{k,i}(+)) = \left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_{k,i}(+)} \quad \text{(e.15)}$$

The iterative procedure to calculate $\hat{\mathbf{x}}_k(+)$, K_k and $P_k(+)$ is given by:

$$\hat{\mathbf{x}}_{k,i+1}(+) = \hat{\mathbf{x}}_k(-) + K_{k,i}[\mathbf{z}_k - \mathbf{h}_k(\mathbf{x}_{k,i}(+)) - H_k(\mathbf{x}_k(-) - \hat{\mathbf{x}}_{k,i}(+))] \quad \text{(e.16)}$$

$$K_{k,i} = P_{k,i}(+)H_k^T(\hat{\mathbf{x}}_{k,i}(+))[H_k(\hat{\mathbf{x}}_{k,i}(+))P_{k,i}(+)H_k^T(\hat{\mathbf{x}}_{k,i}(+)) + R_k]^{-1} \quad \text{(e.17)}$$

$$P_{k,i+1}(+) = [I - K_{k,i}H_k(\hat{\mathbf{x}}_{k,i}(+))]P_{k,i}(+) \quad \text{(e.18)}$$

with the initial conditions:

$$P_{k,0}(+) = P_k(-) \quad \text{(e.19)}$$

$$\hat{\mathbf{x}}_{k,0}(+) = \hat{\mathbf{x}}_k(-) \quad \text{(e.20)}$$

As many iterations are performed as long as modifications of the estimate are obtained.
As evident from eq. (e.14), the linearisation is made with respect to the measurement equation. A similar procedure can be devised for iterating over the non-linear dynamic model when available.

APPENDIX F

Haupt's two step non-linear recursive iterative estimator

The usual models are supposed to be available:

$$\mathbf{z}_k = \mathbf{F}(\mathbf{x}_k, t_k) + \mathbf{v}_k \qquad \text{measurement model} \qquad \text{(f.1)}$$

$$\mathbf{x}_{k+1} = \Phi_k \cdot \mathbf{x}_k + \Gamma_k \cdot \mathbf{w}_k \qquad \text{dynamics model} \qquad \text{(f.2)}$$

Measurement and process noise are assumed to be white, Gaussian, zero mean with covariance matrix respectively R_k and Q_k.
The non-linearity is at first cancelled out by defining a new first step state vector:

$$\mathbf{y}_k = \mathbf{f}_k(\mathbf{x}_k) \qquad \text{(f.3)}$$

The vector function $\mathbf{f}$ has to be chosen in order to make eq. (f.1) linear in the new first step state vector:

$$\mathbf{z}_k = H_k \mathbf{y}_k + \mathbf{v}_k \qquad \text{(f.4)}$$

Initial conditions $\mathbf{y}_0, P_0$ are given.
The two steps will be analysed sequentially.

First step
The measurement update is given as usual by:

$$\mathbf{y}_k(+) = \mathbf{y}_k(-) + P_{yk}(+) H_k^T R_k^{-1} (\mathbf{z}_k - H_k \cdot \mathbf{y}_k(-)) \qquad \text{(f.5)}$$

$$P_{yk}(+) = [P_{yk}^{-1}(-) + H_k^T R_k^{-1} H_k]^{-1} \qquad \text{(f.6)}$$

The time update is found by expanding the vector function $\mathbf{f}$ and by treating the first step state as a perturbation term (Haupt et al., 1995). The time update results:

$$\mathbf{y}_{k+1}(-) = \mathbf{y}_k(+) + \mathbf{f}_{k+1}(\mathbf{x}_{k+1}(-)) - \mathbf{f}_k(\mathbf{x}_k(+)) \qquad \text{(f.7)}$$

$$P_{yk+1}(-) = P_{yk}(+) + \left.\frac{\partial \mathbf{f}_{k+1}}{\partial \mathbf{x}_{k+1}}\right|_{\mathbf{x}_{k+1}=\mathbf{x}_{k+1}(-)} \cdot P_{xk+1}(-) \cdot \left.\frac{\partial \mathbf{f}_{k+1}}{\partial \mathbf{x}_{k+1}}\right|^T_{\mathbf{x}_{k+1}=\mathbf{x}_{k+1}(-)} + \left.\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\mathbf{x}_k(-)} \cdot P_{xk}(+) \cdot \left.\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k}\right|^T_{\mathbf{x}_k=\mathbf{x}_k(-)} \qquad \text{(f.8)}$$

Second step
The measurement update is:

$$\hat{\mathbf{x}}_{ki+1} = \hat{\mathbf{x}}_{ki} - H_{Gki}^{-1} \cdot \mathbf{q}_{ki}^{T} \tag{f.9}$$

$$P_{xki}(+) = H_{Gki}^{-1} \tag{f.10}$$

where the Hessian matrix is approximated according to Gauss' method:

$$H_{Gki} = \left.\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\hat{\mathbf{x}}_{ki}}^{T} \cdot P_{yk}^{-1}(+) \cdot \left.\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\hat{\mathbf{x}}_{ki}} \tag{f.11}$$

and the gradient of the cost function is approximated in the following way:

$$\mathbf{q}_i = \left.\frac{\partial J_x}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_i} = -[\mathbf{z}_k(+) - \mathbf{f}_k(\hat{\mathbf{x}}_{ki})]^T P_{yk}^{-1}(+) \left.\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\hat{\mathbf{x}}_{ki}} \tag{f.12}$$

The time update is:

$$\mathbf{x}_{k+1}(-) = \Phi_k \cdot \hat{\mathbf{x}}_k \tag{f.13}$$

$$P_{xk+1}(-) = \Phi_k P_{xk} \Phi_k^T + \Gamma_k Q_k \Gamma_k^T \tag{f.14}$$

It is noted that the second step optimisation must be carried out between the measurement and the time updates of the first time optimisation at each time step for problems in which the time variation is not separable. In the case of separable time variation, the first step optimisation reduces to the static KF and the second step need not be done at every time step, as the two steps are decoupled.

APPENDIX G

Adaptation of the MME estimator to gas turbine diagnostics

The Hamiltonian function is defined:

$$H = H(\mathbf{x},\dot{\mathbf{b}},t) = [N(\mathbf{x})\cdot\dot{\mathbf{b}}]^T \cdot W \cdot [N(\mathbf{x})\cdot\dot{\mathbf{b}}] + \boldsymbol{\lambda}^T \cdot [\mathbf{f}(\mathbf{x},t) + N(\mathbf{x})\cdot\dot{\mathbf{b}}] \tag{g.1}$$

It is noted that H is here called Hamiltonian even though the term should be used only when the parameters H is function of are of the order zero, whereas $\dot{\mathbf{b}}$ is included in (g.1).
By integration, the modified cost function can be written in terms of the Hamiltonian function as follows:

$$\bar{J} = \sum_{k=1}^{M} [\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k]^T \cdot R^{-1} \cdot [\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k] + \int_{t_0}^{t_M} [H(\mathbf{x},\dot{\mathbf{b}},t) + \dot{\boldsymbol{\lambda}}^T \cdot \mathbf{x}]dt - [\boldsymbol{\lambda}^T \cdot \mathbf{x}]_{t_0}^{t_M} \tag{g.2}$$

For $\bar{J}$ to be minimum, its variation has to be zero for every possible variation of its variables. The variation of the constrained cost function is:

$$\delta\bar{J} = \sum_{k=1}^{M} \delta\varphi(\mathbf{x}_k, \mathbf{b}_k) + \int_{t_o}^{t_M} [\frac{\partial H}{\partial \mathbf{x}} \cdot \delta\mathbf{x} + \frac{\partial H}{\partial \dot{\mathbf{b}}} \cdot \delta\dot{\mathbf{b}} + \dot{\boldsymbol{\lambda}}^T \cdot \delta\mathbf{x}]dt - [\boldsymbol{\lambda}^T \cdot \delta\mathbf{x}]_{t_0}^{t_M} \tag{g.3}$$

where:

$$\varphi(\mathbf{x}_k,\mathbf{b}_k) = [\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k]^T \cdot R^{-1} \cdot [\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k] \tag{g.4}$$

The bias variation can be written:

$$\delta\mathbf{b}(t) = \int_{t_0}^{t_M} \delta\dot{\mathbf{b}}(t)dt + \delta\mathbf{b}_0 \tag{g.5}$$

The variation of the terms in the summation is:

$$\delta\varphi(\mathbf{x}_k,\mathbf{b}_k) = -2[\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k]^T \cdot R^{-1} \cdot \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]_k \delta\mathbf{x}_k +$$
$$-2[\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k]^T \cdot R^{-1} \cdot [\int_{t_0}^{t_M} \delta\dot{\mathbf{b}}dt + \delta\mathbf{b}_0] \tag{g.6}$$

The following initial conditions are assumed:

$$\mathbf{x}(t_0) = \mathbf{x}_0 = 0 \tag{g.7}$$

$$\mathbf{b}(t_0) = \mathbf{b}_0 = 0 \tag{g.8}$$

A piecewise constant function is defined:

$$\mathbf{p}_k = \begin{cases} -2[\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k] & \text{if } 0 \le t \le t_k \\ 0 & \text{if } t > t_k \end{cases} \tag{g.9}$$

The variation of the modified cost function then becomes:

$$\delta\bar{J} = \sum_{k=1}^{M} -2[\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k]^T \cdot R^{-1} \cdot \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]_k \cdot \delta\mathbf{x}_k + \sum_{k=1}^{M} \mathbf{p}_k^T \cdot \delta\mathbf{b}_0 +$$
$$+ \int_{t_0}^{t_M} [(\frac{\partial H}{\partial \mathbf{x}} + \dot{\boldsymbol{\lambda}}^T) \cdot \delta\mathbf{x} + (\frac{\partial H}{\partial \dot{\mathbf{b}}} + \mathbf{p}_k^T) \cdot \delta\dot{\mathbf{b}}]dt - [\boldsymbol{\lambda}^T \cdot \delta\mathbf{x}]_{t_0}^{t_M} \tag{g.10}$$

Since the above variation has to be zeroed for every possible combination of variations of the variables, the following Two Point Boundary Value Problem has to be solved:

$$\frac{\partial H}{\partial \mathbf{x}} = -\dot{\boldsymbol{\lambda}}^T \tag{g.11}$$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},t) + N(\mathbf{x}) \cdot \dot{\mathbf{b}} \tag{g.12}$$

$$\frac{\partial H}{\partial \dot{\mathbf{b}}} = -\mathbf{p}_k^T \tag{g.13}$$

with the conditions:

$$\mathbf{x}_0 = 0 \tag{g.14}$$

$$\mathbf{b}_0 = 0 \tag{g.15}$$

$$\boldsymbol{\lambda}^T(t_k^+) = \boldsymbol{\lambda}^T(t_k^-) + 2R^{-1} \cdot \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]_k \cdot [\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k) - \mathbf{b}_k] \tag{g.16}$$

Since the Hamiltonian function is actually given by:

$$H(\mathbf{x},\dot{\mathbf{b}},t)=\left[\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-1}\cdot\dot{\mathbf{b}}\right]^{T}\cdot W\cdot\left[\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-1}\cdot\dot{\mathbf{b}}\right]+\boldsymbol{\lambda}^{T}\cdot\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-1}\cdot(\dot{\mathbf{y}}-\dot{\mathbf{b}}) \tag{g.17}$$

equations (g.11)-(g.13) become respectively:

$$2\left[\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-1}\cdot\dot{\mathbf{b}}\right]^{T}\cdot W\cdot\left[\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-2}\cdot\left[\frac{\partial^{2}\mathbf{h}}{\partial \mathbf{x}^{2}}\right]\cdot\dot{\mathbf{b}}\right]+\boldsymbol{\lambda}^{T}\cdot\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-2}\cdot\left[\frac{\partial^{2}\mathbf{h}}{\partial \mathbf{x}^{2}}\right]\cdot(\dot{\mathbf{y}}-\dot{\mathbf{b}})=\dot{\boldsymbol{\lambda}}^{T} \tag{g.18}$$

$$\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]\cdot\dot{\mathbf{x}}=\dot{\mathbf{y}}-\dot{\mathbf{b}} \tag{g.19}$$

$$2W\cdot\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{-1}\cdot\dot{\mathbf{b}}-\boldsymbol{\lambda}=-\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{T}\mathbf{p}_{k} \tag{g.20}$$

If the weight matrix W is assumed not only symmetric but also diagonal, eq. (g.20) can be further simplified:

$$\dot{\mathbf{b}}=\frac{1}{2}W^{-1}\cdot\left(\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{T}\cdot\boldsymbol{\lambda}-\left(\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^{T}\right)^{2}\cdot\mathbf{p}_{k}\right) \tag{g.21}$$

APPENDIX H

Delta-delta learning algorithm

The 4 heuristics introduced in section 3.2.1 can be realised by introducing an objective function formally equivalent to the one defined in (3.3):

$$\varepsilon(n) = \frac{1}{2}\sum_{j=1}^{p} e_j^2(n) \tag{h.1}$$

but where $\varepsilon(n)$ is a function of the learning rate parameters of the various weights ($\eta_{ji}(n)$).
A steepest descent technique is then applied in the space of the learning parameters:

$$\Delta\eta_{ji}(n+1) = -\gamma \cdot \frac{\partial\varepsilon(n)}{\partial\eta_{ji}(n)} \tag{h.2}$$

where the constant γ is the control step-size parameter for the learning rate adaptation procedure.
The partial derivative in the R.H.S. of (h.2) can be calculated as follows:

$$\frac{\partial\varepsilon(n)}{\partial\eta_{ji}(n)} = \frac{\partial\varepsilon(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial\eta_{ji}(n)} \tag{h.3}$$

The first two factors in the R.H.S. of (h.3) can be calculated as follows:

$$\frac{\partial\varepsilon(n)}{\partial y_j(n)} = -e_j(n) \tag{h.4}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \tag{h.5}$$

The third factor can be calculated through the definition of $v_j(n)$:

$$v_j(n) = \sum_{i=0}^{N} y_i(n) \cdot w_{ji}(n) \tag{h.6}$$

The weight can be written as follows:

$$w_{ji}(n) = w_{ji}(n-1) - \eta_{ji}(n) \cdot \frac{\partial E(n-1)}{\partial w_{ji}(n-1)} \tag{h.7}$$

According to (h.6) and (h.7):

$$\frac{\partial v_j(n)}{\partial \eta_{ji}(n)} = -y_i(n) \cdot \frac{\partial E(n-1)}{\partial w_{ji}(n-1)} \tag{h.8}$$

(h.3) then becomes:

$$\frac{\partial \varepsilon(n)}{\partial \eta_{ji}(n)} = \varphi_j'(v_j(n)) \cdot e_j(n) \cdot y_i(n) \cdot \frac{\partial E(n-1)}{\partial w_{ji}(n-1)} \tag{h.9}$$

According to (3.4) and (3.12):

$$\frac{\partial \varepsilon(n)}{\partial \eta_{ji}(n)} = -\frac{\partial E(n)}{\partial w_{ji}(n)} \cdot \frac{\partial E(n-1)}{\partial w_{ji}(n-1)} \tag{h.10}$$

and then:

$$\Delta \eta_{ji}(n+1) = \gamma \cdot \frac{\partial E(n)}{\partial w_{ji}(n)} \cdot \frac{\partial E(n-1)}{\partial w_{ji}(n-1)} \tag{h.11}$$

The training algorithm outlined above, though, is affected by the following drawback: if the derivative of the error surface with respect to a particular weight has the same sign but small magnitudes at two consecutive iterations, the positive adjustment applied to the learning rate for that weight is very small. On the other hand, if the derivative of the error surface with respect to a weight has opposite signs and large magnitudes at two consecutive iterations, the negative adjustment applied to that weight will be very large. Under these circumstances, it is very difficult to choose an appropriate value for γ.

Delta-bar-delta learning algorithm

This learning algorithm allows to circumvent the problem affecting the delta-delta algorithm described above.
If the following definition is given:

$$D_{ji}(n) = \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{h.12}$$

and $S_{ji}(n)$ is defined as an exponentially weighted sum of the current and past derivatives of the error surface with respect to w_{ji} :

$$S_{ji}(n) = (1-\xi)\cdot D_{ji}(n-1) + \xi \cdot S_{ji}(n-1) \tag{h.13}$$

where ξ is a positive constant, the learning parameter adaptation process is defined:

$$\Delta\eta_{ji}(n+1) = \begin{cases} k & \text{if } S_{ji}(n-1)\cdot D_{ji}(n) > 0 \\ -\beta\cdot\eta_{ji}(n) & \text{if } S_{ji}(n-1)\cdot D_{ji}(n) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{h.14}$$

Basically, the learning rate parameter $\eta_{ji}(n)$ is incremented linearly but decremented exponentially. A linear increase prevents the learning rate parameter from growing too fast, whereas an exponential decrease means that it remains positive and decreases rapidly.

APPENDIX I

Test samples from the NN-based Sensor Failure Detection and Isolation system

In the sequel some examples of the typical SFDI are given for different number and choices of sensor and engine faults.
In the printouts the following definitions apply:

- Y is the calculated output vector
- D is the desired or target vector
- X is the input vector.

The quantity of interest for SFDI is the XY weighted RMS, which is highlighted for convenience.
Measurement deltas are in the following order:

$$\Delta W_{1A}, \Delta P_{13}, \Delta T_{13}, \Delta P_{21}, \Delta T_{21}, \Delta W_{21}, \Delta P_3, \Delta T_3, \Delta T_5, \Delta P_5, \Delta F, \Delta N_H, \Delta N_L$$

The combination of sensors reported is the one providing the smallest WRMS.
The DY error is not used for SFDI but is reported to check the actual estimation errors.
It is worth noting that:

- measurement noise is strongly reduced
- even accommodation is possible in some cases.

```
------------------------------minimum WRMS-----------------------------
engine faults
ΔΓHPC     ΔηHPC
-2.25     -2.50
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  3  5
actual input vector:
  -0.8491  -0.5015   1.9761  -0.7517   1.9170  -2.3573  -1.8308   0.4213   1.5659  -0.7509  -1.1114   0.3347  -0.6480
corrected input vector:
  -0.8491  -0.5015  -0.0154  -0.7517  -0.1216  -2.3573  -1.8308   0.4213   1.5659  -0.7509  -1.1114   0.3347  -0.6480
desired output vector:
  -1.2330  -0.4461   0.0161  -0.6661  -0.1298  -2.4315  -1.7253   0.5178   1.5685  -0.8355  -1.2041   0.2314  -0.6150
corrected output vector:
  -1.1824  -0.4326  -0.0585  -0.6759  -0.2047  -2.4260  -1.7367   0.4746   1.5896  -0.8440  -1.2189   0.3135  -0.6120
 ------XY------
absolute errors:
   0.3333   0.0689   0.0431   0.0757   0.0831   0.0687   0.0941   0.0533   0.0237   0.0931   0.1074   0.0212   0.0361
ABD:
   0.0847
absolute weighted errors:
   4.8493   1.2280   0.2699   1.4797   0.5008   1.0485   1.2997   0.5193   0.4335   1.7693   1.4309   0.5542   0.3690
weighted ABD:
   1.2117
squared errors:
   0.1111   0.0047   0.0019   0.0057   0.0069   0.0047   0.0089   0.0028   0.0006   0.0087   0.0115   0.0004   0.0013
RMS:
   0.1141
weighted squared errors:
  23.5159   1.5080   0.0729   2.1894   0.2508   1.0994   1.6892   0.2696   0.1880   3.1303   2.0474   0.3071   0.1362
weighted RMS:
   1.6734<<<<<<<<<<
 ------DY------
absolute errors:
   0.0506   0.0135   0.0746   0.0098   0.0749   0.0056   0.0114   0.0432   0.0211   0.0085   0.0148   0.0821   0.0030
ABD:
   0.0318
absolute weighted errors:
   0.7356   0.2405   0.4668   0.1914   0.4516   0.0848   0.1571   0.4213   0.3858   0.1623   0.1970   2.1494   0.0307
weighted ABD:
   0.4365
squared errors:
   0.0026   0.0002   0.0056   0.0001   0.0056   0.0000   0.0001   0.0019   0.0004   0.0001   0.0002   0.0067   0.0000
RMS:
   0.0425
weighted squared errors:
   0.5411   0.0579   0.2179   0.0366   0.2039   0.0072   0.0247   0.1775   0.1489   0.0263   0.0388   4.6201   0.0009
weighted RMS:
   0.6851

------------------------------minimum WRMS-----------------------------
engine faults
ΔΓHPC     ΔηHPC
-2.00     -2.63
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  1  2
actual input vector:
   0.8319   1.6173  -0.2154  -0.5185  -0.5689  -2.2596  -1.6235   0.1855   1.2714  -0.8000  -1.2174  -0.0067  -0.7161
corrected input vector:
  -1.0556  -0.3887  -0.2154  -0.5185  -0.5689  -2.2596  -1.6235   0.1855   1.2714  -0.8000  -1.2174  -0.0067  -0.7161
desired output vector:
  -1.0171  -0.3914  -0.3128  -0.6065  -0.4505  -2.2002  -1.6287   0.2322   1.2825  -0.7711  -1.0722  -0.0288  -0.7178
corrected output vector:
  -1.0564  -0.3843  -0.0586  -0.6045  -0.1922  -2.1627  -1.5559   0.4117   1.4015  -0.7516  -1.0763   0.0073  -0.5548
 ------XY------
absolute errors:
   0.0008   0.0044   0.1568   0.0861   0.3767   0.0970   0.0676   0.2263   0.1301   0.0483   0.1412   0.0140   0.1613
ABD:
   0.1162
absolute weighted errors:
   0.0120   0.0793   0.9817   1.6817   2.2710   1.4801   0.9343   2.2039   2.3820   0.9187   1.8804   0.3653   1.6498
weighted ABD:
   1.2954
squared errors:
   0.0000   0.0000   0.0246   0.0074   0.1419   0.0094   0.0046   0.0512   0.0169   0.0023   0.0199   0.0002   0.0260
RMS:
   0.1530
weighted squared errors:
   0.0001   0.0063   0.9637   2.8281   5.1575   2.1908   0.8729   4.8574   5.6739   0.8440   3.5359   0.1334   2.7218
weighted RMS:
   1.5137<<<<<<<<<<
 ------DY------
absolute errors:
   0.0393   0.0072   0.2542   0.0020   0.2584   0.0375   0.0728   0.1795   0.1190   0.0194   0.0040   0.0361   0.1630
ABD:
   0.0917
absolute weighted errors:
   0.5725   0.1279   1.5911   0.0387   1.5575   0.5725   1.0056   1.7488   2.1784   0.3696   0.0535   0.9449   1.6668
weighted ABD:
   0.9560
squared errors:
   0.0015   0.0001   0.0646   0.0000   0.0667   0.0014   0.0053   0.0322   0.0142   0.0004   0.0000   0.0013   0.0266
RMS:
   0.1284
weighted squared errors:
   0.3278   0.0164   2.5316   0.0015   2.4258   0.3277   1.0113   3.0584   4.7455   0.1366   0.0029   0.8929   2.7781
weighted RMS:
   1.1850
```

```
--------------------------------minimum WRMS----------------------------
engine faults:
ΔΓHPT     ΔηHPT
2.6250  -0.7500
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  4  6  8 10
actual input vector:
  -1.4671  -0.5117  -0.0371   1.4254  -0.2326  -0.3150  -2.5169   1.9639   1.3145   1.2520  -1.0005  -0.9040  -0.6676
corrected input vector:
  -1.4671  -0.5117  -0.0371  -0.5992  -0.2326  -2.1701  -2.5169  -0.0752   1.3145  -0.7456  -1.0005  -0.9040  -0.6676
desired output vector:
  -1.0914  -0.4257  -0.1120  -0.6351  -0.2490  -2.2128  -2.2678   0.0246   1.3311  -0.7912  -1.1260  -0.8730  -0.6295
corrected output vector:
  -1.1043  -0.3856  -0.1105  -0.5837  -0.2401  -2.1715  -2.5442  -0.1190   1.3970  -0.7487  -1.0659  -0.9489  -0.6294
 ------XY------
absolute errors:
   0.3627   0.1261   0.0734   0.0155   0.0075   0.0014   0.0273   0.0438   0.0825   0.0031   0.0654   0.0449   0.0383
ABD:
   0.0686
absolute weighted errors:
   4.9763   2.2929   0.5273   0.2969   0.0516   0.0153   0.1652   1.0675   1.5303   0.0563   0.8462   0.5167   0.4510
weighted ABD:
   0.9841
squared errors:
   0.1316   0.0159   0.0054   0.0002   0.0001   0.0000   0.0007   0.0019   0.0068   0.0000   0.0043   0.0020   0.0015
RMS:
   0.1145
weighted squared errors:
  24.7638   5.2572   0.2780   0.0881   0.0027   0.0002   0.0273   1.1395   2.3419   0.0032   0.7161   0.2669   0.2034
weighted RMS:
   1.6429<<<<<<<<<<
 ------DY------
absolute errors:
   0.0129   0.0401   0.0015   0.0514   0.0089   0.0413   0.2764   0.1437   0.0660   0.0425   0.0601   0.0759   0.0002
ABD:
   0.0631
absolute weighted errors:
   0.1770   0.7286   0.0110   0.9875   0.0614   0.4617   1.6729   3.5006   1.2227   0.7789   0.7777   0.8727   0.0019
weighted ABD:
   0.8657
squared errors:
   0.0002   0.0016   0.0000   0.0026   0.0001   0.0017   0.0764   0.0206   0.0043   0.0018   0.0036   0.0058   0.0000
RMS:
   0.0956
weighted squared errors:
   0.0313   0.5308   0.0001   0.9752   0.0038   0.2132   2.7985  12.2541   1.4950   0.6067   0.6049   0.7617   0.0000
weighted RMS:
   1.2489


--------------------------------minimum WRMS----------------------------
engine faults:
ΔΓHPT     ΔηHPT
-2.63     -2.50
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  4  6  8 10
actual input vector:
  -0.2025  -0.2297   0.2092   1.6844  -0.0612   0.8779   1.3790   3.3513   0.6053   1.5661  -0.4041  -0.0190  -0.2785
corrected input vector:
  -0.2025  -0.2297   0.2092  -0.2435  -0.0612  -0.8065   1.3790   1.1649   0.6053  -0.3275  -0.4041  -0.0190  -0.2785
desired output vector:
  -0.5661  -0.2346   0.0741  -0.3375   0.0061  -1.0343   1.3962   1.3044   0.6403  -0.3888  -0.5547  -0.0175  -0.2344
corrected output vector:
  -0.5279  -0.1789   0.0122  -0.2583  -0.0319  -0.8022   1.4210   1.2193   0.5918  -0.3202  -0.4648  -0.0985  -0.2354
 ------XY------
absolute errors:
   0.3254   0.0508   0.1970   0.0147   0.0294   0.0043   0.0420   0.0544   0.0135   0.0073   0.0607   0.0795   0.0431
ABD:
   0.0709
absolute weighted errors:
   4.4634   0.9238   1.4151   0.2829   0.2023   0.0477   0.2545   1.3254   0.2505   0.1333   0.7860   0.9142   0.5077
weighted ABD:
   0.8851
squared errors:
   0.1059   0.0026   0.0388   0.0002   0.0009   0.0000   0.0018   0.0030   0.0002   0.0001   0.0037   0.0063   0.0019
RMS:
   0.1127
weighted squared errors:
  19.9220   0.8535   2.0024   0.0800   0.0409   0.0023   0.0647   1.7567   0.0628   0.0178   0.6177   0.8358   0.2577
weighted RMS:
   1.4281<<<<<<<<<<
 ------DY------
absolute errors:
   0.0383   0.0558   0.0619   0.0792   0.0380   0.2321   0.0248   0.0851   0.0485   0.0686   0.0899   0.0810   0.0010
ABD:
   0.0695
absolute weighted errors:
   0.5251   1.0144   0.4446   1.5222   0.2614   2.5923   0.1501   2.0730   0.8987   1.2564   1.1639   0.9318   0.0118
weighted ABD:
   0.9881
squared errors:
   0.0015   0.0031   0.0038   0.0063   0.0014   0.0539   0.0006   0.0072   0.0023   0.0047   0.0081   0.0066   0.0000
RMS:
   0.0875
weighted squared errors:
   0.2757   1.0289   0.1977   2.3171   0.0683   6.7198   0.0225   4.2973   0.8077   1.5785   1.3547   0.8683   0.0001
weighted RMS:
   1.2259
```

```
--------------------------------minimum WRMS----------------------------
engine faults:
ΔηFAN     ΔΓFAN     ΔΓLPT     ΔηLPT
-1.20     3.00      -1.80     -3.00
>>>>>>>>>>>>>>>>>>minimisation by varying measurements 12 13
actual input vector:
  -1.5840  -0.7605   0.3896  -0.1072  -0.1875  -1.6854  -1.1244  -0.6196   0.9041  -0.7322  -1.2579   1.2058   0.8616
corrected input vector:
  -1.5840  -0.7605   0.3896  -0.1072  -0.1875  -1.6854  -1.1244  -0.6196   0.9041  -0.7322  -1.2579  -0.8717  -1.3086
desired output vector:
  -1.1124  -0.6989   0.1704  -0.1365  -0.1554  -1.3946  -1.1458  -0.5679   0.8984  -0.7526  -1.0965  -0.8135  -1.2022
corrected output vector:
  -1.1785  -0.7685   0.2053  -0.0485  -0.1876  -1.3713  -1.1434  -0.5913   0.9186  -0.7904  -1.1621  -0.8565  -1.3060
 ------XY------
absolute errors:
   0.4055   0.0080   0.1843   0.0587   0.0001   0.3142   0.0189   0.0282   0.0145   0.0582   0.0958   0.0152   0.0025
ABD:
   0.0926
absolute weighted errors:
   6.5254   0.0717   0.9189   0.4889   0.0008   3.4377   0.2479   0.2440   0.2438   0.9413   1.0333   0.3143   0.0573
weighted ABD:
   1.1173
squared errors:
   0.1645   0.0001   0.0340   0.0034   0.0000   0.0987   0.0004   0.0008   0.0002   0.0034   0.0092   0.0002   0.0000
RMS:
   0.1556
weighted squared errors:
  42.5803   0.0051   0.8444   0.2390   0.0000  11.8175   0.0615   0.0595   0.0595   0.8861   1.0678   0.0988   0.0033
weighted RMS:
   2.1072<<<<<<<<<<
 ------DY------
absolute errors:
   0.0661   0.0696   0.0349   0.0880   0.0322   0.0234   0.0024   0.0234   0.0201   0.0377   0.0656   0.0430   0.1039
ABD:
   0.0470
absolute weighted errors:
   1.0638   0.6263   0.1741   0.7328   0.2078   0.2558   0.0317   0.2026   0.3397   0.6100   0.7075   0.8900   2.3378
weighted ABD:
   0.6292
squared errors:
   0.0044   0.0048   0.0012   0.0077   0.0010   0.0005   0.0000   0.0005   0.0004   0.0014   0.0043   0.0019   0.0108
RMS:
   0.0548
weighted squared errors:
   1.1316   0.3922   0.0303   0.5371   0.0432   0.0654   0.0010   0.0411   0.1154   0.3721   0.5005   0.7920   5.4652
weighted RMS:
   0.8543<<<<<<<<<<

--------------------------------minimum WRMS----------------------------
engine faults:
ΔηFAN     ΔΓFAN     ΔΓLPT     ΔηLPT
-3.00     1.80      -0.60     -1.20
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  1  2
actual input vector:
   1.1085   1.1701   0.6963   1.5114   0.2506  -0.1097   0.2438  -0.1428  -0.2074  -0.3661  -0.6171  -0.4106  -0.4373
corrected input vector:
  -0.5739  -0.6350   0.6963   1.5114   0.2506  -0.1097   0.2438  -0.1428  -0.2074  -0.3661  -0.6171  -0.4106  -0.4373
desired output vector:
  -0.6584  -0.7923   0.5610   1.4778   0.2872   0.3624   0.2587  -0.1540  -0.1554  -0.3993  -0.6291  -0.4105  -0.3867
corrected output vector:
  -0.5742  -0.7390   0.4821   1.4572   0.1939   0.4142   0.2445  -0.1800  -0.2307  -0.3805  -0.5923  -0.4232  -0.4258
 ------XY------
absolute errors:
   0.0003   0.1040   0.2142   0.0541   0.0567   0.5240   0.0007   0.0372   0.0234   0.0143   0.0248   0.0126   0.0115
ABD:
   0.0829
absolute weighted errors:
   0.0052   0.9353   1.0680   0.4510   0.3659   5.7334   0.0085   0.3214   0.3947   0.2313   0.2678   0.2609   0.2587
weighted ABD:
   0.7925
squared errors:
   0.0000   0.0108   0.0459   0.0029   0.0032   0.2745   0.0000   0.0014   0.0005   0.0002   0.0006   0.0002   0.0001
RMS:
   0.1618
weighted squared errors:
   0.0000   0.8748   1.1407   0.2034   0.1339  32.8718   0.0001   0.1033   0.1558   0.0535   0.0717   0.0680   0.0669
weighted RMS:
   1.6582<<<<<<<<<<
 ------DY------
absolute errors:
   0.0842   0.0533   0.0789   0.0206   0.0934   0.0518   0.0142   0.0260   0.0753   0.0189   0.0368   0.0128   0.0391
ABD:
   0.0466
absolute weighted errors:
   1.3552   0.4791   0.3935   0.1712   0.6020   0.5671   0.1867   0.2252   1.2711   0.3054   0.3966   0.2647   0.8802
weighted ABD:
   0.5460
squared errors:
   0.0071   0.0028   0.0062   0.0004   0.0087   0.0027   0.0002   0.0007   0.0057   0.0004   0.0014   0.0002   0.0015
RMS:
   0.0540
weighted squared errors:
   1.8367   0.2296   0.1549   0.0293   0.3624   0.3216   0.0349   0.0507   1.6157   0.0933   0.1573   0.0701   0.7748
weighted RMS:
   0.6640
```

```
------------------------------minimum WRMS-----------------------------
engine faults:
ΔηFAN     ΔΓFAN     ΔΓLPT     ΔηLPT
-3.00     1.80      -0.60     -1.20
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  1  3  5  7
actual input vector:
   1.1085  -0.8299   2.6963   1.5114   2.2506  -0.1097   2.2438  -0.1428  -0.2074  -0.3661  -0.6171  -0.4106  -0.4373
corrected input vector:
  -0.6097  -0.8299   0.3386   1.5114   0.1950  -0.1097   0.1588  -0.1428  -0.2074  -0.3661  -0.6171  -0.4106  -0.4373
desired output vector:
  -0.6584  -0.7923   0.5610   1.4778   0.2872   0.3624   0.2587  -0.1540  -0.1554  -0.3993  -0.6291  -0.4105  -0.3867
corrected output vector:
  -0.6099  -0.7715   0.4732   1.4089   0.1889   0.3963   0.2323  -0.1717  -0.2113  -0.4029  -0.6258  -0.4103  -0.4292
 ------XY------
absolute errors:
   0.0002   0.0583   0.1346   0.1025   0.0061   0.5061   0.0734   0.0289   0.0039   0.0368   0.0087   0.0003   0.0081
ABD:
   0.0745
absolute weighted errors:
   0.0031   0.5248   0.6708   0.8535   0.0393   5.5376   0.9627   0.2499   0.0661   0.5948   0.0936   0.0065   0.1818
weighted ABD:
   0.7527
squared errors:
   0.0000   0.0034   0.0181   0.0105   0.0000   0.2561   0.0054   0.0008   0.0000   0.0014   0.0001   0.0000   0.0001
RMS:
   0.1509
weighted squared errors:
   0.0000   0.2754   0.4500   0.7285   0.0015  30.6656   0.9268   0.0624   0.0044   0.3538   0.0088   0.0000   0.0330
weighted RMS:
   1.6055
------DY------
absolute errors:
   0.0485   0.0207   0.0878   0.0689   0.0983   0.0339   0.0264   0.0178   0.0559   0.0036   0.0033   0.0001   0.0425
ABD:
   0.0391
absolute weighted errors:
   0.7812   0.1866   0.4378   0.5737   0.6337   0.3714   0.3466   0.1536   0.9425   0.0581   0.0351   0.0027   0.9571
weighted ABD:
   0.4215
squared errors:
   0.0024   0.0004   0.0077   0.0047   0.0097   0.0012   0.0007   0.0003   0.0031   0.0000   0.0000   0.0000   0.0018
RMS:
   0.0496
weighted squared errors:
   0.6103   0.0348   0.1917   0.3291   0.4016   0.1379   0.1201   0.0236   0.8883   0.0034   0.0012   0.0000   0.9161
weighted RMS:
   0.5305

------------------------------minimum WRMS-----------------------------
engine faults:
ΔηFAN     ΔΓFAN     ΔΓLPT     ΔηLPT
-3.00     3.00      -0.60     0.00
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  1  3  5  7
actual input vector:
   1.7976  -0.3511   2.5189   2.0699   2.5055   0.7827   2.5112  -0.2016  -0.4282  -0.1649  -0.0128  -0.4988  -0.3367
corrected input vector:
  -0.1459  -0.3511   0.6052   2.0699   0.2653   0.7827   0.4336  -0.2016  -0.4282  -0.1649  -0.0128  -0.4988  -0.3367
desired output vector:
  -0.0858  -0.2654   0.6528   1.9481   0.3429   0.6254   0.4378  -0.1904  -0.4476  -0.0478  -0.1038  -0.4883  -0.3888
corrected output vector:
  -0.1467  -0.3594   0.5420   1.8976   0.2422   0.6492   0.4139  -0.2263  -0.4620  -0.1079  -0.1878  -0.4978  -0.3164
 ------XY------
absolute errors:
   0.0008   0.0083   0.0632   0.1723   0.0230   0.1334   0.0197   0.0247   0.0338   0.0570   0.1750   0.0010   0.0203
ABD:
   0.0564
absolute weighted errors:
   0.0127   0.0746   0.3152   1.4349   0.1485   1.4600   0.2585   0.2136   0.5700   0.9208   1.8873   0.0206   0.4577
weighted ABD:
   0.5980
squared errors:
   0.0000   0.0001   0.0040   0.0297   0.0005   0.0178   0.0004   0.0006   0.0011   0.0032   0.0306   0.0000   0.0004
RMS:
   0.0825
weighted squared errors:
   0.0002   0.0056   0.0993   2.0591   0.0221   2.1316   0.0668   0.0456   0.3249   0.8478   3.5620   0.0004   0.2095
weighted RMS:
   0.8492<<<<<<<<<<
------DY------
absolute errors:
   0.0609   0.0940   0.1108   0.0505   0.1006   0.0239   0.0239   0.0359   0.0144   0.0601   0.0840   0.0095   0.0724
ABD:
   0.0570
absolute weighted errors:
   0.9802   0.8458   0.5525   0.4207   0.6487   0.2614   0.3135   0.3101   0.2431   0.9722   0.9056   0.1965   1.6290
weighted ABD:
   0.6369
squared errors:
   0.0037   0.0088   0.0123   0.0026   0.0101   0.0006   0.0006   0.0013   0.0002   0.0036   0.0071   0.0001   0.0052
RMS:
   0.0657
weighted squared errors:
   0.9607   0.7154   0.3053   0.1770   0.4208   0.0683   0.0983   0.0962   0.0591   0.9452   0.8201   0.0386   2.6537
weighted RMS:
   0.7524
```

```
--------------------------------minimum WRMS-----------------------------
engine faults:
ΔΓHPT     ΔηHPT     ΔCd
-1.50     -1.50     -1.13
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  4  5  6  7
actual input vector:
  -1.2214   0.0190  -0.0884   1.8117   1.9312   1.7271   2.6596  -0.0097   0.5034   0.3244  -0.4192  -0.3566  -0.4363
corrected input vector:
  -1.2214   0.0190  -0.0884  -0.1551  -0.0964  -0.7076   0.1526  -0.0097   0.5034   0.3244  -0.4192  -0.3566  -0.4363
desired output vector:
  -1.0431   0.0446  -0.0048  -0.2007  -0.1502  -0.7417   0.6288  -0.0312   0.4981   0.4076  -0.5531  -0.3353  -0.5130
corrected output vector:
  -0.9441   0.0536   0.0601  -0.1547  -0.0726  -0.7052   0.1386  -0.0550   0.5183   0.3414  -0.4666  -0.2846  -0.4049
 ------XY------
absolute errors:
   0.2773   0.0346   0.1484   0.0004   0.0238   0.0024   0.0140   0.0452   0.0149   0.0169   0.0474   0.0720   0.0314
ABD:
   0.0561
absolute weighted errors:
   1.2316   0.7009   0.9535   0.0076   0.1667   0.0223   0.0665   0.4207   0.2030   0.1154   0.4937   0.6993   0.7396
weighted ABD:
   0.4478
squared errors:
   0.0769   0.0012   0.0220   0.0000   0.0006   0.0000   0.0002   0.0020   0.0002   0.0003   0.0022   0.0052   0.0010
RMS:
   0.0928
weighted squared errors:
   1.5170   0.4912   0.9091   0.0001   0.0278   0.0005   0.0044   0.1770   0.0412   0.0133   0.2438   0.4890   0.5470
weighted RMS:
   0.5858<<<<<<<<<<
 ------DY------
absolute errors:
   0.0990   0.0091   0.0649   0.0460   0.0777   0.0365   0.4903   0.0238   0.0202   0.0662   0.0865   0.0506   0.1080
ABD:
   0.0907
absolute weighted errors:
   0.4398   0.1832   0.4170   0.9186   0.5434   0.3413   2.3200   0.2213   0.2754   0.4510   0.9006   0.4917   2.5448
weighted ABD:
   0.7729
squared errors:
   0.0098   0.0001   0.0042   0.0021   0.0060   0.0013   0.2404   0.0006   0.0004   0.0044   0.0075   0.0026   0.0117
RMS:
   0.1496
weighted squared errors:
   0.1934   0.0336   0.1739   0.8439   0.2953   0.1165   5.3825   0.0490   0.0758   0.2034   0.8111   0.2418   6.4758
weighted RMS:
   1.0704

--------------------------------minimum WRMS-----------------------------
engine faults:
ΔΓHPT     ΔηHPT     ΔCd
3.00      -1.50     -2.26
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  1  5  9 13
actual input vector:
  -1.1515  -0.0856  -0.0823  -0.7411   1.8594  -3.3883  -4.6125  -1.5888   3.9417   0.5026  -1.7438  -1.5419   0.7147
corrected input vector:
  -2.8525  -0.0856  -0.0823  -0.7411  -0.4812  -3.3883  -4.6125  -1.5888   2.3075   0.5026  -1.7438  -1.5419  -1.4091
desired output vector:
  -2.7720  -0.1346   0.0299  -0.6012  -0.2698  -2.7920  -4.5920  -1.5202   1.9537   0.3857  -1.7473  -1.5614  -1.2809
corrected output vector:
  -2.8485  -0.1403  -0.0756  -0.6455  -0.3828  -3.0730  -4.4529  -1.6219   2.1378   0.3282  -1.8069  -1.7282  -1.3798
 ------XY------
absolute errors:
   0.0041   0.0547   0.0067   0.0956   0.0985   0.3153   0.1596   0.0331   0.1697   0.1744   0.0631   0.1863   0.0293
ABD:
   0.1070
absolute weighted errors:
   0.0181   1.1076   0.0428   1.9095   0.6890   2.9485   0.7554   0.3083   2.3089   1.1883   0.6572   1.8092   0.6912
weighted ABD:
   1.1103
squared errors:
   0.0000   0.0030   0.0000   0.0091   0.0097   0.0994   0.0255   0.0011   0.0288   0.0304   0.0040   0.0347   0.0009
RMS:
   0.1377
weighted squared errors:
   0.0003   1.2267   0.0018   3.6461   0.4747   8.6937   0.5706   0.0950   5.3308   1.4120   0.4320   3.2731   0.4777
weighted RMS:
   1.4042<<<<<<<<<<
 ------DY------
absolute errors:
   0.0765   0.0057   0.1055   0.0443   0.1129   0.2810   0.1392   0.1018   0.1841   0.0574   0.0596   0.1668   0.0988
ABD:
   0.1103
absolute weighted errors:
   0.3396   0.1148   0.6779   0.8853   0.7901   2.6280   0.6585   0.9469   2.5050   0.3914   0.6207   1.6198   2.3287
weighted ABD:
   1.1159
squared errors:
   0.0058   0.0000   0.0111   0.0020   0.0127   0.0790   0.0194   0.0104   0.0339   0.0033   0.0036   0.0278   0.0098
RMS:
   0.1297
weighted squared errors:
   0.1153   0.0132   0.4596   0.7837   0.6242   6.9063   0.4337   0.8966   6.2749   0.1532   0.3853   2.6239   5.4228
weighted RMS:
   1.3893
```

```
------------------------------MINIMUM WRMS----------------------------
engine faults:
ΔΓHPT     ΔηHPT     ΔΓLPT     ΔηLPT
1.50      -2.25     -3.00     -1.50
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  2  8  9 10
actual input vector:
  -1.7134   1.9314  -0.0700  -0.4701  -0.2552  -5.1730  -5.1610   0.0786   5.0774   0.6831  -1.7332  -2.7765  -0.8827
corrected input vector:
  -1.7134  -0.1027  -0.0700  -0.4701  -0.2552  -5.1730  -5.1610  -2.0353   3.0920  -1.2699  -1.7332  -2.7765  -0.8827
desired output vector:
  -1.8122  -0.1346  -0.0529  -0.4744  -0.2656  -5.0521  -5.1590  -2.0064   3.1228  -1.3041  -1.7869  -2.7434  -0.8795
corrected output vector:
  -1.8073  -0.1117  -0.0386  -0.4720  -0.2629  -5.0539  -5.2061  -2.0181   3.0701  -1.2747  -1.7722  -2.7576  -0.8687
 ------XY------
absolute errors:
   0.0938   0.0089   0.0314   0.0019   0.0078   0.1191   0.0451   0.0173   0.0219   0.0048   0.0389   0.0189   0.0141
ABD:
   0.0326
absolute weighted errors:
   0.5103   0.1472   0.6113   0.0301   0.1473   0.8529   0.4207   0.3744   0.4837   0.1066   0.5682   0.3926   0.3434
weighted ABD:
   0.3838
squared errors:
   0.0088   0.0001   0.0010   0.0000   0.0001   0.0142   0.0020   0.0003   0.0005   0.0000   0.0015   0.0004   0.0002
RMS:
   0.0472
weighted squared errors:
   0.2604   0.0217   0.3737   0.0009   0.0217   0.7274   0.1770   0.1402   0.2339   0.0114   0.3229   0.1541   0.1179
weighted RMS:
   0.4440<<<<<<<<<<
 ------DY------
absolute errors:
   0.0050   0.0230   0.0143   0.0024   0.0026   0.0018   0.0471   0.0117   0.0526   0.0294   0.0148   0.0142   0.0108
ABD:
   0.0177
absolute weighted errors:
   0.0270   0.3793   0.2789   0.0381   0.0502   0.0130   0.4393   0.2541   1.1645   0.6564   0.2159   0.2946   0.2633
weighted ABD:
   0.3134
squared errors:
   0.0000   0.0005   0.0002   0.0000   0.0000   0.0000   0.0022   0.0001   0.0028   0.0009   0.0002   0.0002   0.0001
RMS:
   0.0237
weighted squared errors:
   0.0007   0.1439   0.0778   0.0015   0.0025   0.0002   0.1930   0.0646   1.3561   0.4309   0.0466   0.0868   0.0693
weighted RMS:
   0.4362

------------------------------MINIMUM WRMS----------------------------
engine faults:
ΔΓHPT     ΔηHPT     ΔΓLPT     ΔηLPT
-1.50     -3.00     3.00      -2.25
>>>>>>>>>>>>>>>>>>minimisation by varying measurements  2  8  9 10
actual input vector:
  -1.8713  -0.1514  -0.6821  -2.2904  -0.6890  -0.9804   0.6060   1.9924   2.9382   0.6479  -2.0150   0.1971  -1.4979
corrected input vector:
  -1.8713  -2.0112  -0.6821  -2.2904  -0.6890  -0.9804   0.6060   0.0552   0.9668  -1.3809  -2.0150   0.1971  -1.4979
desired output vector:
  -2.1219  -2.0850  -0.6370  -2.3100  -0.8361  -0.8639   0.5323   0.0529   0.8826  -1.3993  -2.1147   0.1898  -1.5282
corrected output vector:
  -2.0804  -2.0131  -0.5954  -2.2008  -0.7843  -0.9097   0.5297   0.0784   0.9169  -1.3654  -2.0462   0.1936  -1.4382
 ------XY------
absolute errors:
   0.2090   0.0019   0.0867   0.0896   0.0952   0.0707   0.0763   0.0233   0.0499   0.0155   0.0312   0.0036   0.0597
ABD:
   0.0625
absolute weighted errors:
   1.1370   0.0320   1.6887   1.4278   1.8062   0.5062   0.7114   0.5052   1.1029   0.3454   0.4551   0.0738   1.4586
weighted ABD:
   0.8654
squared errors:
   0.0437   0.0000   0.0075   0.0080   0.0091   0.0050   0.0058   0.0005   0.0025   0.0002   0.0010   0.0000   0.0036
RMS:
   0.0818
weighted squared errors:
   1.2927   0.0010   2.8518   2.0385   3.2623   0.2562   0.5060   0.2553   1.2164   0.1193   0.2071   0.0055   2.1276
weighted RMS:
   1.0429<<<<<<<<<<
 ------DY------
absolute errors:
   0.0416   0.0718   0.0416   0.1093   0.0518   0.0458   0.0025   0.0255   0.0343   0.0339   0.0685   0.0037   0.0900
ABD:
   0.0477
absolute weighted errors:
   0.2260   1.1859   0.8108   1.7407   0.9821   0.3279   0.0236   0.5534   0.7588   0.7560   0.9994   0.0776   2.1981
weighted ABD:
   0.8185
squared errors:
   0.0017   0.0052   0.0017   0.0119   0.0027   0.0021   0.0000   0.0007   0.0012   0.0011   0.0047   0.0000   0.0081
RMS:
   0.0562
weighted squared errors:
   0.0511   1.4064   0.6574   3.0302   0.9646   0.1075   0.0006   0.3062   0.5757   0.5715   0.9988   0.0060   4.8318
weighted RMS:
   1.0193
```

APPENDIX J

Test samples from the EP-based diagnostic system applied to the EJ200

In the sequel results from a number of test cases are shown.
For correct interpretation of the printout, the following information is necessary:

- "actual" refers to the fault that has been implanted
- "best" refers to the best solution found so far
- "current" refers to the best solution in the current population of strings
- a positive value of the delta means a decrease of the corresponding performance parameter
- "obmean" is the mean value of the objective function over the population
- "online p." and "offline p." are the running average of the mean and best strings. They are useful to keep track of the convergence of the whole population
- "stdev" is the standard deviation of the corresponding performance parameter extended to the entire population
- the integer numbers under the line "fault classes" are the number of strings for the various fault classes
- the three environment and power setting parameters are biased as well. They are fuel flow (g/s), ambient pressure (kPa) and ambient temperature (K) respectively
- the quantities under "environment and power setting parameters' accommodation" are the errors of estimation for those parameters
- "ABD" is the absolute deviation
- the final three columns are the values of the terms to be summed up in the objective function. Depending on the number of biases assumed to be present, either the 2 or the 4 largest values are discarded as biased measurements. Those measurements are underlined
- often the optimiser is stopped before reaching complete comvergence on the basis of the low value of the standard deviations of the performance parameters and the concentration on a single fault class
- The order of measurements is as follows:

  W1a
  P13
  T13
  P21
  T21
  W21
  P3
  T3
  T5
  P5
  F
  Nhp
  Nlp

a. 1 faulty component, 2 biases

```
iter.          actual        best        current     stdev
-------------------------------------------------------------------

                objf         objf         objf
  15/ 100       9.282        6.368        6.368
 nstrings: 4048
     obmean   0.00 DWfan   0.00 DWfan   0.00 DWfan    1.39
   117.2094   0.00 DEfout  0.00 DEfout  0.00 DEfout   0.65
  online p.  -3.00 DWfan  -2.96 DWfan  -2.96 DWfan    1.05
   221.1762   2.00 DEfin   2.16 DEfin   2.16 DEfin    0.66
 offline p.   0.00 DWhpc   0.00 DWhpc   0.00 DWhpc    0.08
    14.3201   0.00 DEhpc   0.00 DEhpc   0.00 DEhpc    0.06
              0.00 DWhpt   0.00 DWhpt   0.00 DWhpt    0.02
              0.00 DEhpt   0.00 DEhpt   0.00 DEhpt    0.05
              0.00 DWlpt   0.00 DWlpt   0.00 DWlpt    0.05
              0.00 DElpt   0.00 DElpt   0.00 DElpt    0.06
              0.00 DCa     0.00 DCa     0.00 DCa      0.19
 environment and power setting parameters
              747.5198     747.4049     747.4049
               86.5036      86.6195      86.6195
              263.7045     263.8304     263.8304
 fault classes
   0 1837    0    0    1    0 2154    1    0    0    0
   4    4    2   45    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                           0.11         0.11
                           0.12         0.12
                           0.13         0.13
                           RMS          RMS
                           0.05         0.05
                           ABD          ABD
                           0.02         0.02
                    6.29         6.37         6.37
                   11.68        11.53        11.53
                    0.67         0.81         0.81
                    0.74         0.92         0.92
                    0.15         0.23         0.23
                    1.11         0.79         0.79
                    0.98         0.04         0.04
                    0.03         0.56         0.56
                    1.57         0.56         0.56
                    0.91         0.69         0.69
                    0.84         0.18         0.18
                    0.12         0.83         0.83
                    2.17         0.76         0.76
```

```
iter.          actual        best        current     stdev
-------------------------------------------------------------------

                objf         objf         objf
  22/ 200       7.955        5.041        5.722
 nstrings: 4048
     obmean   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.04
   123.0151   0.00 DEfout  0.00 DEfout  0.00 DEfout   0.04
  online p.   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.02
   189.1197   0.00 DEfin   0.00 DEfin   0.00 DEfin    0.02
 offline p.   0.00 DWhpc   0.00 DWhpc   0.00 DWhpc    0.40
    13.8448   0.00 DEhpc   0.00 DEhpc   0.00 DEhpc    0.59
              3.00 DWhpt   2.83 DWhpt   2.95 DWhpt    1.23
              3.00 DEhpt   2.93 DEhpt   2.96 DEhpt    0.96
              0.00 DWlpt   0.00 DWlpt   0.00 DWlpt    0.12
              0.00 DElpt   0.00 DElpt   0.00 DElpt    0.13
              0.00 DCa     0.00 DCa     0.00 DCa      0.16
 environment and power setting parameters
              747.0969     748.1353     747.6235
               83.7102      83.5430      83.4520
              264.3195     264.0201     264.0201
 fault classes
   2    0  403 3474   15    7    0    1    0    0    0
   1    1    0    0    0    5    0    7  128    4
 environment and power setting parameters' accommodation
```

```
                                1.04          0.53
                                0.17          0.26
                                0.30          0.30
                                RMS           RMS
                                0.06          0.02
                                ABD           ABD
                                0.02          0.01
                        0.16          0.06          0.31
                        0.53          0.27          1.27
                        0.35          0.27          0.18
                        0.66          1.02          0.06
                       13.03         13.47         13.34
                        6.03          5.79          5.98
                        0.50          0.17          0.33
                        0.54          0.52          0.39
                        0.56          0.02          0.07
                        0.04          0.65          0.26
                        2.55          0.64          1.40
                        0.49          0.61          0.07
                        1.58          0.79          1.37

iter.          actual          best          current       stdev
-------------------------------------------------------------------------

                       objf          objf          objf
  31/ 100             8.328         7.147         7.252
 nstrings: 4048
      obmean    0.00 DWfan    0.00 DWfan    0.00 DWfan    0.00
      53.0574   0.00 DEfout   0.00 DEfout   0.00 DEfout   0.00
   online p.    0.00 DWfan    0.00 DWfan    0.00 DWfan    0.00
     152.9499   0.00 DEfin    0.00 DEfin    0.00 DEfin    0.00
  offline p.    0.00 DWhpc    0.00 DWhpc    0.00 DWhpc    0.00
      14.1711   0.00 DEhpc    0.00 DEhpc    0.00 DEhpc    0.00
                0.00 DWhpt    0.00 DWhpt    0.00 DWhpt    0.00
                0.00 DEhpt    0.00 DEhpt    0.00 DEhpt    0.00
               -3.00 DWlpt   -3.00 DWlpt   -3.00 DWlpt    0.44
                3.00 DElpt    2.99 DElpt    2.95 DElpt    0.24
                0.00 DCa      0.00 DCa      0.00 DCa      0.00
 environment and power setting parameters
                 811.2414      810.8183      810.5399
                  83.6934       83.7414       83.8072
                 263.9430      263.9655      264.1133
 fault classes
    0    0    0    0 4047    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    1
 environment and power setting parameters' accommodation
                                0.42          0.70
                                0.05          0.11
                                0.02          0.17
                                RMS           RMS
                                0.00          0.02
                                ABD           ABD
                                0.00          0.01
                        0.41          0.45          0.47
                        1.55          1.54          1.70
                        0.38          0.48          0.02
                        0.06          0.06          0.11
                       12.60         12.74         12.31
                        6.30          6.31          6.35
                        0.68          0.46          0.26
                        0.79          0.34          0.69
                        0.89          0.15          0.30
                        1.76          1.87          1.85
                        0.11          0.53          0.90
                        0.53          0.98          0.91
                        1.16          0.29          0.05

iter.               actual          best          current       stdev
-------------------------------------------------------------------------

                       objf          objf          objf
  63/ 100             6.418         6.516         6.686
 nstrings: 4048
      obmean    0.00 DWfan    0.00 DWfan    0.00 DWfan    0.00
      12.6782   0.00 DEfout   0.00 DEfout   0.00 DEfout   0.00
   online p.    0.00 DWfan    0.00 DWfan    0.00 DWfan    0.00
      79.7442   0.00 DEfin    0.00 DEfin    0.00 DEfin    0.00
```

```
offline p.  -2.00 DWhpc   -2.08 DWhpc   -2.07 DWhpc     0.12
     9.8331  3.00 DEhpc    2.96 DEhpc    2.96 DEhpc     0.05
             0.00 DWhpt    0.00 DWhpt    0.00 DWhpt     0.00
             0.00 DEhpt    0.00 DEhpt    0.00 DEhpt     0.00
             0.00 DWlpt    0.00 DWlpt    0.00 DWlpt     0.00
             0.00 DElpt    0.00 DElpt    0.00 DElpt     0.00
             0.00 DCa      0.00 DCa      0.00 DCa       0.00
environment and power setting parameters
              746.8692      747.0269      746.7872
               86.5344       86.5523       86.6563
              312.5873      312.6971      312.9215
fault classes
   0    0 4048    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0
environment and power setting parameters' accommodation
                          0.16          0.08
                          0.02          0.12
                          0.11          0.33
                          RMS           RMS
                          0.03          0.03
                          ABD           ABD
                          0.01          0.01
                  0.50          0.47          0.46
                  0.43          0.23          0.05
                 10.37          9.95          9.45
                 11.60         11.38         11.19
                  0.21          0.22          0.67
                  0.23          0.32          0.25
                  0.03          0.41          0.15
                  0.28          0.70          1.16
                  0.43          0.35          0.06
                  0.68          0.42          0.41
                  2.71          2.96          2.29
                  0.00          0.12          0.11
                  0.91          0.30          1.07
```

b. 1 faulty component, 4 biases

```
iter.        actual        best        current     stdev
-------------------------------------------------------------------------

                objf          objf          objf
  27/ 200       7.197         6.842         8.397
 nstrings: 4048
     obmean   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.42
     92.4513  0.00 DEfout   0.00 DEfout   0.00 DEfout    0.53
  online p.   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
    140.9843  0.00 DEfin    0.00 DEfin    0.00 DEfin     0.00
 offline p.   0.00 DWhpc    0.00 DWhpc    0.00 DWhpc     0.00
     12.3388  0.00 DEhpc    0.00 DEhpc    0.00 DEhpc     0.00
              0.00 DWhpt    1.13 DWhpt    0.93 DWhpt     1.24
              0.00 DEhpt    0.18 DEhpt    0.49 DEhpt     0.62
             -3.00 DWlpt   -2.65 DWlpt   -2.97 DWlpt     0.93
              3.00 DElpt    2.94 DElpt    2.80 DElpt     0.61
              0.00 DCa      0.00 DCa      0.00 DCa       0.04
 environment and power setting parameters
               810.9833      807.5734      809.6642
                83.6799       83.6991       83.6776
               263.4644      263.3889      263.2922
 fault classes
   0    0    0    0  803    0    0    0    0  215    0
   0    0    0    0    0    0    0 3023    0    7
 environment and power setting parameters' accommodation
                          3.41          1.32
                          0.02          0.00
                          0.08          0.17
                          RMS           RMS
                          0.38          0.34
                          ABD           ABD
                          0.17          0.16
                  0.88          0.57          0.65
                  2.09          1.85          2.24
                  0.03          0.47          0.72
                  0.74          0.52          0.26
                 13.99         14.24         14.59
                  4.32          4.26          4.21
```

9.97	0.08	1.12
23.71	21.06	22.52
0.84	11.19	5.87
1.17	2.10	1.32
0.25	1.10	0.09
0.44	0.01	1.26
0.76	0.15	0.73
0.80	0.76	0.93

iter.	actual	best	current	stdev
	objf	objf	objf	
18/ 100	9.914	7.805	7.805	
nstrings: 4048				
obmean	0.00 DWfan	0.00 DWfan	0.00 DWfan	0.05
72.7682	0.00 DEfout	0.00 DEfout	0.00 DEfout	0.01
online p.	0.00 DWfan	0.00 DWfan	0.00 DWfan	0.04
139.1635	0.00 DEfin	0.00 DEfin	0.00 DEfin	0.05
offline p.	0.00 DWhpc	0.00 DWhpc	0.00 DWhpc	0.05
12.8048	0.00 DEhpc	0.00 DEhpc	0.00 DEhpc	0.03
	0.00 DWhpt	0.00 DWhpt	0.00 DWhpt	0.04
	0.00 DEhpt	0.00 DEhpt	0.00 DEhpt	0.02
	-3.00 DWlpt	-2.91 DWlpt	-2.91 DWlpt	0.77
	3.00 DElpt	2.99 DElpt	2.99 DElpt	0.57
	0.00 DCa	0.00 DCa	0.00 DCa	0.04
environment and power setting parameters				
	810.3999	809.8679	809.8679	
	83.6911	83.8059	83.8059	
	264.7235	265.3172	265.3172	
fault classes				
0 0 0	0 4034	0 0 0	0 1	0
0 0 6	0 0	3 0 1	0 3	
environment and power setting parameters' accommodation				
		0.53	0.53	
		0.11	0.11	
		0.59	0.59	
		RMS	RMS	
		0.03	0.03	
		ABD	ABD	
		0.01	0.01	
	0.64	0.89	0.89	
	0.03	0.10	0.10	
	1.62	0.62	0.62	
	1.90	1.79	1.79	
	3.08	0.95	0.95	
	0.57	0.11	0.11	
	1.48	2.83	2.83	
	19.91	17.91	17.91	
	33.47	30.39	30.39	
	13.41	13.97	13.97	
	11.09	12.56	12.56	
	0.23	0.17	0.17	
	0.35	0.34	0.34	

iter.	actual	best	current	stdev
	objf	objf	objf	
22/ 100	6.156	5.490	5.559	
nstrings: 4048				
obmean	-3.00 DWfan	-2.91 DWfan	-2.93 DWfan	0.76
62.6175	3.00 DEfout	2.92 DEfout	2.95 DEfout	0.47
online p.	0.00 DWfan	0.00 DWfan	0.00 DWfan	0.08
138.8873	0.00 DEfin	0.00 DEfin	0.00 DEfin	0.03
offline p.	0.00 DWhpc	0.00 DWhpc	0.00 DWhpc	0.04
11.9454	0.00 DEhpc	0.00 DEhpc	0.00 DEhpc	0.03
	0.00 DWhpt	0.00 DWhpt	0.00 DWhpt	0.05
	0.00 DEhpt	0.00 DEhpt	0.00 DEhpt	0.03
	0.00 DWlpt	0.00 DWlpt	0.00 DWlpt	0.11
	0.00 DElpt	0.00 DElpt	0.00 DElpt	0.18
	0.00 DCa	0.00 DCa	0.00 DCa	0.03
environment and power setting parameters				
	746.2067	746.0188	745.7188	

```
                86.5154       86.5500       86.4559
               263.8365      263.7739      263.6775
 fault classes
 3995    0    0    0    3    0    6    0    1   40    2
    0    0    0    0    0    1    0    0    0    0
 environment and power setting parameters' accommodation
                              0.19          0.49
                              0.03          0.06
                              0.06          0.16
                              RMS           RMS
                              0.04          0.03
                              ABD           ABD
                              0.02          0.01
                     0.18          0.08          0.23
                     1.06          1.36          0.59
                     0.07          0.69          0.86
                     1.11          0.70          0.26
                    14.80         15.32         15.51
                     5.59          5.59          5.67
                    11.08         11.25         11.64
                    23.56         24.26         24.54
                     0.95          0.27          0.23
                     1.80          1.67          2.31
                     0.19          0.18          0.16
                     0.28          0.09          0.27
                     0.51          0.45          0.65

iter.         actual         best         current      stdev
-------------------------------------------------------------------------

                    objf          objf          objf
   15/ 100          8.388         6.977         6.977
 nstrings: 4048 .
       obmean   -3.00 DWfan   -2.98 DWfan   -2.98 DWfan     1.08
      78.8779    3.00 DEfout    2.98 DEfout    2.98 DEfout    0.88
   online p.     0.00 DWfan     0.00 DWfan     0.00 DWfan     0.54
     146.9547    0.00 DEfin     0.00 DEfin     0.00 DEfin     0.27
  offline p.     0.00 DWhpc     0.00 DWhpc     0.00 DWhpc     0.08
      14.5909    0.00 DEhpc     0.00 DEhpc     0.00 DEhpc     0.05
                 0.00 DWhpt     0.00 DWhpt     0.00 DWhpt     0.08
                 0.00 DEhpt     0.00 DEhpt     0.00 DEhpt     0.04
                 0.00 DWlpt     0.00 DWlpt     0.00 DWlpt     0.11
                 0.00 DElpt     0.00 DElpt     0.00 DElpt     0.15
                 0.00 DCa       0.00 DCa       0.00 DCa       0.28
 environment and power setting parameters
                811.9957      812.9502      812.9502
                 86.5409       86.7041       86.7041
                311.9326      311.9447      311.9447
 fault classes
 3644   27    3    0   20   62  212    1    1    5   42
    0    1    0   11    1    1    2    0    5   10
 environment and power setting parameters' accommodation
                              0.95          0.95
                              0.16          0.16
                              0.01          0.01
                              RMS           RMS
                              0.01          0.01
                              ABD           ABD
                              0.00          0.00
                     6.75          6.27          6.27
                    11.67          9.82          9.82
                    11.72         11.87         11.87
                    12.71         11.04         11.04
                     1.89          1.74          1.74
                     1.24          1.66          1.66
                     1.48          0.10          0.10
                     0.05          0.23          0.23
                     0.51          0.08          0.08
                     0.17          1.56          1.56
                     1.11          0.39          0.39
                     1.44          1.21          1.21
                     0.50          0.01          0.01
```

c. 2 faulty components, 2 biases

```
iter.          actual         best         current      stdev
-------------------------------------------------------------------------
```

```
                 objf          objf          objf
  70/ 200        7.951         6.999         8.565
 nstrings: 4048
      obmean   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
      51.5480  0.00 DEfout   0.00 DEfout   0.00 DEfout    0.00
   online p.  -2.00 DWfan   -1.82 DWfan   -2.19 DWfan     0.49
     134.1437  2.00 DEfin    2.09 DEfin    2.07 DEfin     0.25
  offline p.   0.00 DWhpc    0.00 DWhpc    0.00 DWhpc     0.00
      15.6885  0.00 DEhpc    0.00 DEhpc    0.00 DEhpc     0.00
               0.00 DWhpt    0.00 DWhpt    0.00 DWhpt     0.00
               0.00 DEhpt    0.00 DEhpt    0.00 DEhpt     0.00
               3.00 DWlpt    3.00 DWlpt    3.00 DWlpt     0.40
               3.00 DElpt    2.99 DElpt    2.87 DElpt     0.31
               0.00 DCa      0.00 DCa      0.00 DCa       0.00
 environment and power setting parameters
              810.1086      810.0928      810.8972
               86.5057       86.2715       86.0382
              263.7532      263.2512      263.0554
 fault classes
    0    0    0    0    0    0    0    0    0    0    0
    0    0 4048    0    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                            0.02          0.79
                            0.23          0.47
                            0.50          0.70
                            RMS           RMS
                            0.06          0.08
                            ABD           ABD
                            0.03          0.04
                   1.02          0.90          0.96
                   0.97          0.08          0.44
                   1.65          0.03          0.16
                   0.08          1.67          1.74
                  14.10         15.47         15.56
                   4.61          4.65          4.56
                   0.37          0.08          0.68
                   1.51          0.11          0.36
                   0.20          1.05          0.39
                   0.45          1.09          0.67
                   0.13          0.32          2.30
                   0.25          0.32          0.10
                   1.32          1.33          0.75

iter.            actual        best          current      stdev
-------------------------------------------------------------------------

                 objf          objf          objf
  200/ 200       7.741         9.801         9.989
 nstrings: 4048
      obmean   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
      12.1162  0.00 DEfout   0.00 DEfout   0.00 DEfout    0.00
   online p.   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
      62.9034  0.00 DEfin    0.00 DEfin    0.00 DEfin     0.00
  offline p.  -3.00 DWhpc   -3.00 DWhpc   -2.92 DWhpc     0.04
      15.2029  3.00 DEhpc    2.59 DEhpc    2.47 DEhpc     0.03
               2.00 DWhpt    2.11 DWhpt    2.10 DWhpt     0.03
               1.00 DEhpt    1.41 DEhpt    1.56 DEhpt     0.02
               0.00 DWlpt    0.00 DWlpt    0.00 DWlpt     0.00
               0.00 DElpt    0.00 DElpt    0.00 DElpt     0.00
               0.00 DCa      0.00 DCa      0.00 DCa       0.00
 environment and power setting parameters
              747.2620      746.8628      746.7428
               86.5601       86.5682       86.5220
              312.4150      312.5468      312.4087
 fault classes
    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0 4048    0    0    0    0    0
 environment and power setting parameters' accommodation
                            0.40          0.52
                            0.01          0.04
                            0.13          0.01
                            RMS           RMS
                            0.19          0.25
                            ABD           ABD
                            0.09          0.13
                   0.05          0.16          0.24
```

```
                  0.70          0.32          0.02
                 12.55         12.18         12.64
                 12.18         12.54         12.90
                  0.43          0.08          0.53
                  1.38          1.48          1.62
                  0.17          0.68          0.03
                  0.67          3.53          5.57
                  0.27          0.34          0.02
                  1.43          1.01          0.64
                  0.36          1.03          1.28
                  1.88          0.77          0.03
                  0.40          0.41          0.01

iter.        actual        best        current     stdev
-------------------------------------------------------------------

                objjf         objjf         objjf
  18/ 100       7.019         5.582         5.582
 nstrings: 4048
     obmean   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.39
     90.9668  0.00 DEfout  0.00 DEfout  0.00 DEfout   0.17
  online p.   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.00
    220.3144  0.00 DEfin   0.00 DEfin   0.00 DEfin    0.00
 offline p.   0.00 DWhpc   0.00 DWhpc   0.00 DWhpc    0.00
     16.3424  0.00 DEhpc   0.00 DEhpc   0.00 DEhpc    0.00
              0.00 DWhpt   0.00 DWhpt   0.00 DWhpt    0.00
              0.00 DEhpt   0.00 DEhpt   0.00 DEhpt    0.00
             -3.00 DWlpt  -2.98 DWlpt  -2.99 DWlpt    0.74
              3.00 DElpt   2.97 DElpt   2.47 DElpt    0.52
              3.00 DCa     2.95 DCa     2.93 DCa      0.67
 environment and power setting parameters
             746.3024     746.0060     745.4938
              86.5172      86.5328      86.1796
             264.0024     263.9447     263.1999
 fault classes
   0    0    0    0    1    0    0    0    0  115    0
   0    0    0    0    0    0    0    0    0 3932
 environment and power setting parameters' accommodation
                          0.30         0.81
                          0.02         0.34
                          0.06         0.80
                          RMS          RMS
                          0.02         0.17
                          ABD          ABD
                          0.01         0.06
                  0.09         0.25         0.25
                  0.96         0.94         0.94
                  0.45         0.78         0.78
                  0.58         0.54         0.54
                  0.30         0.00         0.00
                  0.07         0.05         0.05
                  0.30         0.04         0.04
                  2.14         1.42         1.42
                 33.82        34.93        34.93
                 11.35        11.76        11.76
                  0.64         0.65         0.65
                  0.73         0.02         0.02
                  0.76         0.88         0.88

iter.          actual        best        current     stdev
-------------------------------------------------------------------

                objjf         objjf         objjf
 200/ 200       7.807         7.968         9.024
 nstrings: 4048
     obmean   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.00
     10.8553  0.00 DEfout  0.00 DEfout  0.00 DEfout   0.00
  online p.  -2.00 DWfan  -1.92 DWfan  -2.37 DWfan    0.05
     57.1736  2.00 DEfin   1.98 DEfin   1.00 DEfin    0.04
 offline p.   0.00 DWhpc   0.00 DWhpc   0.00 DWhpc    0.00
     11.8934  0.00 DEhpc   0.00 DEhpc   0.00 DEhpc    0.00
             -1.00 DWhpt  -1.18 DWhpt  -1.49 DWhpt    0.03
              3.00 DEhpt   2.92 DEhpt   2.43 DEhpt    0.03
              0.00 DWlpt   0.00 DWlpt   0.00 DWlpt    0.00
              0.00 DElpt   0.00 DElpt   0.00 DElpt    0.00
              0.00 DCa     0.00 DCa     0.00 DCa      0.00
 environment and power setting parameters
```

```
                811.7169      812.2031      808.5258
                 86.5443       86.7716       86.7043
                311.9104      312.2670      312.8423
 fault classes
    0    0    0    0    0    0    0    0    0    0    0
    0 4048    0    0    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                              0.49          3.19
                              0.23          0.16
                              0.36          0.93
                              RMS           RMS
                              0.07          0.41
                              ABD           ABD
                              0.04          0.24
                     0.03          0.23          0.65
                     0.06          1.36          0.07
                    12.37         11.50          9.39
                    11.94         10.35          1.40
                     1.20          0.34          1.67
                     0.04          0.13          1.75
                     0.48          0.38          0.08
                     0.63          0.84          0.05
                     0.24          0.48         18.83
                     2.17          1.06          1.22
                     1.54          1.72          1.86
                     0.80          0.84          0.00
                     0.61          0.60          0.26

                     0.71          0.72          0.82
```

d. 2 faulty components, 4 biases

```
iter.              actual        best        current     stdev
-------------------------------------------------------------------------

                   objf          objf          objf
  82/ 200          7.116         1.180         1.480
 nstrings: 4048
      obmean    0.00 DWfan    0.00 DWfan    0.00 DWfan    0.00
      16.3375   0.00 DEfout   0.00 DEfout   0.00 DEfout   0.00
   online p.   -3.00 DWfan   -2.71 DWfan   -2.74 DWfan    0.22
      71.6519   1.00 DEfin    0.84 DEfin    0.83 DEfin    0.11
  offline p.    0.00 DWhpc    0.00 DWhpc    0.00 DWhpc    0.00
       6.0430   0.00 DEhpc    0.00 DEhpc    0.00 DEhpc    0.00
                0.00 DWhpt    0.00 DWhpt    0.00 DWhpt    0.00
                0.00 DEhpt    0.00 DEhpt    0.00 DEhpt    0.00
               -1.00 DWlpt   -1.27 DWlpt   -1.29 DWlpt    0.18
                3.00 DElpt    2.89 DElpt    2.79 DElpt    0.13
                0.00 DCa      0.00 DCa      0.00 DCa      0.00
 environment and power setting parameters
                811.7596      809.8394      809.4850
                 86.5345       86.7319       86.7301
                311.9372      312.4292      312.4292
 fault classes
    0    0    0    0    0    0    0    0    0    0    0
    0    0 4048    0    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                              1.92          2.27
                              0.20          0.20
                              0.49          0.49
                              RMS           RMS
                              0.14          0.15
                              ABD           ABD
                              0.08          0.09
                     0.25          0.12          0.18
                     1.45          0.01          0.11
                     0.83          0.10          0.07
                     0.26          0.51          0.69
                     0.87          0.00          0.03
                     0.51          0.21          0.19
                     0.26          0.03          0.12
                    19.93         18.33         18.41
                    30.24         35.39         36.56
                     8.81          9.76          9.78
                    10.39         13.11         13.11
                     1.99          0.16          0.07
```

```
                0.71          0.04          0.03


iter.        actual        best        current     stdev
------------------------------------------------------------------------

                objf          objf          objf
  49/ 100       5.889         3.763         3.965
 nstrings: 4048
     obmean   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
     15.0755  0.00 DEfout   0.00 DEfout   0.00 DEfout    0.00
   online p.  0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
     79.7860  0.00 DEfin    0.00 DEfin    0.00 DEfin     0.00
  offline p.  0.00 DWhpc    0.00 DWhpc    0.00 DWhpc     0.00
      8.2780  0.00 DEhpc    0.00 DEhpc    0.00 DEhpc     0.00
              0.00 DWhpt    0.00 DWhpt    0.00 DWhpt     0.00
              0.00 DEhpt    0.00 DEhpt    0.00 DEhpt     0.00
             -3.00 DWlpt   -3.00 DWlpt   -3.00 DWlpt     0.14
              2.00 DElpt    2.05 DElpt    2.13 DElpt     0.16
              3.00 DCa      2.49 DCa      2.57 DCa       0.21
 environment and power setting parameters
              746.3970      746.5151      746.9205
               83.7102       83.9540       83.8557
              264.3774      264.6136      264.4507
 fault classes
   0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0 4048
 environment and power setting parameters' accommodation
                            0.12          0.52
                            0.24          0.15
                            0.24          0.07
                            RMS           RMS
                            0.16          0.14
                            ABD           ABD
                            0.06          0.06
                  0.70          0.44          0.20
                  1.11          0.67          0.56
                  1.41          1.34          0.96
                  0.29          0.14          0.35
                  0.60          0.39          0.80
                  0.68          0.67          0.63
                  0.12          0.01          0.25
                 22.53         22.24         22.47
                 33.84         34.45         33.60
                 11.39         14.36         13.84
                 11.71         11.22         10.93
                  0.12          0.08          0.14
                  0.85          0.03          0.07

iter.        actual        best        current     stdev
------------------------------------------------------------------------

                objf          objf          objf
 200/ 200       7.005         1.981         1.985
 nstrings: 4048
     obmean   3.00 DWfan    2.96 DWfan    2.95 DWfan     0.01
      2.4268  2.00 DEfout   2.15 DEfout   2.16 DEfout    0.01
   online p.  0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
     39.9040  0.00 DEfin    0.00 DEfin    0.00 DEfin     0.00
  offline p.  0.00 DWhpc    0.00 DWhpc    0.00 DWhpc     0.00
      5.3139  0.00 DEhpc    0.00 DEhpc    0.00 DEhpc     0.00
             -3.00 DWhpt   -1.79 DWhpt   -1.79 DWhpt     0.01
              2.00 DEhpt    2.46 DEhpt    2.46 DEhpt     0.01
              0.00 DWlpt    0.00 DWlpt    0.00 DWlpt     0.00
              0.00 DElpt    0.00 DElpt    0.00 DElpt     0.00
              0.00 DCa      0.00 DCa      0.00 DCa       0.00
 environment and power setting parameters
              746.6335      745.7676      745.8084
               83.7356       83.7420       83.7332
              312.0306      311.6051      311.6036
 fault classes
   0    0    0    0    0    0    0    0 4048    0    0
   0    0    0    0    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                            0.87          0.83
                            0.01          0.00
```

```
                                0.43          0.43
                                RMS           RMS
                                0.41          0.41
                                ABD           ABD
                                0.19          0.19
                      1.07          1.37          1.35
                      0.01          0.20          0.25
                      1.30          0.00          0.05
                      1.09          0.00          0.00
                     12.25         13.69         13.66
                     ------------------------------------
                      5.29          4.67          4.69
                     ------------------------------------
                     12.22          0.00          0.01
                     ------------------------------------
                     21.28         18.57         18.54
                     ------------------------------------
                      0.66          6.88          6.66
                      0.17          0.33          0.30
                      0.19          0.00          0.00
                      1.02          0.01          0.02
                      1.49          0.05          0.01

iter.                 actual         best          current      stdev
-------------------------------------------------------------------------

                      objf          objf          objf
  80/ 200             4.613         9.103        11.641
 nstrings: 4048
      obmean    0.00 DWfan    0.00 DWfan    0.00 DWfan     0.54
      40.5078   0.00 DEfout   0.00 DEfout   0.00 DEfout    0.37
   online p.   -3.00 DWfan   -2.91 DWfan   -2.97 DWfan     0.41
      97.3016   1.00 DEfin    0.98 DEfin    0.98 DEfin     0.16
  offline p.    0.00 DWhpc    0.00 DWhpc    0.00 DWhpc     0.00
      15.4384   0.00 DEhpc    0.00 DEhpc    0.00 DEhpc     0.00
                2.00 DWhpt    2.07 DWhpt    1.95 DWhpt     0.26
                1.00 DEhpt    1.18 DEhpt    0.88 DEhpt     0.13
                0.00 DWlpt    0.00 DWlpt    0.00 DWlpt     0.00
                0.00 DElpt    0.00 DElpt    0.00 DElpt     0.00
                0.00 DCa      0.00 DCa      0.00 DCa       0.47
 environment and power setting parameters
                812.0670      812.3909      812.3909
                 86.4732       86.5959       86.5470
                263.7653      263.7557      263.7551
 fault classes
    0    0    0    0    0    0    3    0    0    0 3958
    0   87    0    0    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                                0.32          0.32
                                0.12          0.07
                                0.01          0.01
                                RMS           RMS
                                0.07          0.04
                                ABD           ABD
                                0.04          0.02
                      5.31          5.22          4.95
                     ------------------------------------
                     11.44         10.76         10.47
                     ------------------------------------
                     14.96         15.32         15.04
                     ------------------------------------
                     13.27         12.33         11.99
                     ------------------------------------
                      0.08          0.43          0.07
                      0.07          0.20          0.42
                      0.44          0.84          1.51
                      0.39          1.62          0.30
                      0.55          1.02          3.13
                      1.78          2.15          2.90
                      0.07          0.25          1.08
                      0.94          1.40          1.75
                      0.28          1.20          0.48
```

APPENDIX K

Test samples from the EP-based diagnostic system applied to the RB199

In the sequel results from a number of test cases are shown.
For correct interpretation of the printout, the following information is necessary:

- "actual" refers to the fault that has been implanted
- "best" refers to the best solution found so far
- "current" refers to the best solution in the current population of strings
- a positive value of the delta means a decrease of the corresponding performance parameter
- "obmean" is the mean value of the objective function over the population
- "online p." and "offline p." are the running average of the mean and best strings. They are useful to keep track of the convergence of the whole population
- "stdev" is the standard deviation of the corresponding performance parameter extended to the entire population
- the integer numbers under the line "fault classes" are the number of strings for the various fault classes
- the three environment and power setting parameters are biased as well. They are fuel flow (g/s), ambient pressure (kPa) and ambient temperature (K) respectively
- the quantities under "environment and power setting parameters' accommodation" are the errors of estimation for those parameters
- "ABD" is the absolute deviation
- the final three columns are the values of the terms to be summed up in the objective function. Depending on the number of biases assumed to be present, either the 2 or the 5 largest values are discarded as biased measurements. Those measurements are underlined
- often the optimiser is stopped before reaching complete comvergence on the basis of the low value of the standard deviations of the performance parameters and the concentration on a single fault class.
- The order of measurements is as follows:

W_{1A}
P_{13}
T_{13}
P_{24}
T_{24}
W_{24}
P_{26}
T_{26}
P_3
T_3
P_{45}
T_{45}

F
N_H
N_I
N_L

a. 1 faulty component, 2 biases

```
iter.        actual        best        current     stdev
-------------------------------------------------------------------------

                 objf         objf         objf
  23/ 100       17.576       16.936       16.936
 nstrings: 3600
     obmean   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.03
   110.4541   0.00 DEfout  0.00 DEfout  0.00 DEfout   0.07
  online p.   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.01
   229.8008   0.00 DEfin   0.00 DEfin   0.00 DEfin    0.02
 offline p.   0.00 DWipc   0.00 DWipc   0.00 DWipc    0.06
    32.5114   0.00 DEipc   0.00 DEipc   0.00 DEipc    0.09
              3.00 DWhpc   2.89 DWhpc   2.89 DWhpc    0.97
              2.00 DEhpc   1.89 DEhpc   1.89 DEhpc    0.53
              0.00 DWhpt   0.00 DWhpt   0.00 DWhpt    0.31
              0.00 DEhpt   0.00 DEhpt   0.00 DEhpt    0.15
              0.00 DWipt   0.00 DWipt   0.00 DWipt    0.05
              0.00 DEipt   0.00 DEipt   0.00 DEipt    0.08
              0.00 DWlpt   0.00 DWlpt   0.00 DWlpt    0.02
              0.00 DElpt   0.00 DElpt   0.00 DElpt    0.05
              0.00 DCa     0.00 DCa     0.00 DCa      0.01
 environment and power setting parameters
             479.8695     479.3734     479.3734
              93.4255      93.5183      93.5183
             256.5222     256.7659     256.7659
 fault classes
   1    0    1 3492    4   17    4    0    0    0    4    0
   0    0    0    0    1    0    0    0    0   10    0    0
   0    0   65    0    0    0    0    0    0    0    1    0
 environment and power setting parameters' accommodation
                          0.50         0.50
                          0.09         0.09
                          0.24         0.24
                          RMS          RMS
                          0.04         0.04
                          ABD          ABD
                          0.02         0.02
                  0.83         0.76         0.76
                  0.23         0.22         0.22
                  3.39         2.83         2.83
                  0.40         0.33         0.33
                  0.66         1.32         1.32
                  1.73         1.83         1.83
                  2.22         3.60         3.60
                  0.88         1.44         1.44
                 11.80        12.35        12.35
                 22.27        21.94        21.94
                  2.59         2.41         2.41
                  0.00         0.49         0.49
                  2.68         1.57         1.57
                  0.03         0.01         0.01
                  0.78         0.09         0.09
                  1.19         0.04         0.04

iter.          actual        best        current     stdev
-------------------------------------------------------------------------

                 objf         objf         objf
  25/ 100       19.157       20.715       20.830
 nstrings: 3600
     obmean   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.00
   113.4792   0.00 DEfout  0.00 DEfout  0.00 DEfout   0.00
  online p.   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.00
   224.0857   0.00 DEfin   0.00 DEfin   0.00 DEfin    0.00
```

```
 offline p.  -2.00 DWipc  -2.44 DWipc  -2.32 DWipc     0.75
     26.7174  3.00 DEipc   2.19 DEipc   2.34 DEipc     0.67
              0.00 DWhpc   0.00 DWhpc   0.00 DWhpc     0.60
              0.00 DEhpc   0.00 DEhpc   0.00 DEhpc     0.53
              0.00 DWhpt   0.00 DWhpt   0.00 DWhpt     0.00
              0.00 DEhpt   0.00 DEhpt   0.00 DEhpt     0.00
              0.00 DWipt   0.00 DWipt   0.00 DWipt     0.03
              0.00 DEipt   0.00 DEipt   0.00 DEipt     0.04
              0.00 DWlpt   0.00 DWlpt   0.00 DWlpt     0.00
              0.00 DElpt   0.00 DElpt   0.00 DElpt     0.00
              0.00 DCa     0.00 DCa     0.00 DCa       0.10
environment and power setting parameters
             520.1361     518.0636     518.2506
              93.4541      93.8261      93.7634
             256.3122     257.9267     257.7207
fault classes
   0    0 2966    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0  565    0    3
   0   66    0    0    0    0    0    0    0    0    0    0
environment and power setting parameters' accommodation
                          2.07         1.89
                          0.37         0.31
                          1.61         1.41
                          RMS          RMS
                          0.25         0.20
                          ABD          ABD
                          0.09         0.07
                  0.74         1.25         1.28
                  0.09         0.02         0.29
                  1.40         6.33         5.64
                  2.61         2.60         2.20
                  0.01         5.37         4.63
                  0.38         0.55         0.71
                  6.94         2.55         4.00
                 15.23        10.27        10.85
                  3.97         2.97         2.51
                  3.79         0.23         0.06
                  0.12         0.44         0.78
                  0.10         1.56         0.55
                  2.92         1.13         1.05
                  0.31         0.12         0.47
                  1.30         0.65         0.44
                  1.40         1.28         1.85

iter.        actual        best        current     stdev
------------------------------------------------------------------------

                objf         objf         objf
   8/ 100      20.017       22.945       22.945
 nstrings: 3600
     obmean   0.00 DWfan   0.00 DWfan   0.00 DWfan     0.25
    219.1823  0.00 DEfout  0.00 DEfout  0.00 DEfout    0.22
  online p.   0.00 DWfan   0.00 DWfan   0.00 DWfan     0.29
    419.8421  0.00 DEfin   0.00 DEfin   0.00 DEfin     0.29
 offline p.   0.00 DWipc   0.00 DWipc   0.00 DWipc     0.29
     32.4787  0.00 DEipc   0.00 DEipc   0.00 DEipc     0.33
              0.00 DWhpc   0.00 DWhpc   0.00 DWhpc     1.19
              0.00 DEhpc   0.00 DEhpc   0.00 DEhpc     1.09
             -3.00 DWhpt  -2.84 DWhpt  -2.84 DWhpt     1.48
              3.00 DEhpt   2.98 DEhpt   2.98 DEhpt     0.90
              0.00 DWipt   0.00 DWipt   0.00 DWipt     0.41
              0.00 DEipt   0.00 DEipt   0.00 DEipt     0.49
              0.00 DWlpt   0.00 DWlpt   0.00 DWlpt     0.32
              0.00 DElpt   0.00 DElpt   0.00 DElpt     0.35
              0.00 DCa     0.00 DCa     0.00 DCa       0.20
 environment and power setting parameters
             479.1929     479.7195     479.7195
              93.4738      94.0233      94.0233
             304.3369     305.0780     305.0780
 fault classes
   3    3    3   23 1326    8    6    1    3    3   11   40
   5    4    0    5   10   70    3    3    1   12   80    3
   4    1 1562   26   10    5  194  100   55   10    5    2
 environment and power setting parameters' accommodation
                          0.53         0.53
                          0.55         0.55
                          0.74         0.74
```

```
                                RMS             RMS
                                0.04            0.04
                                ABD             ABD
                                0.01            0.01
                          1.03          0.34          0.34
                          1.03          4.89          4.89
                          0.52          1.16          1.16
                          2.24          1.89          1.89
                          2.09          0.32          0.32
                          2.81          3.38          3.38
                          1.16          1.41          1.41
                          0.42          2.12          2.12
                          4.89          0.86          0.86
                          1.39          1.65          1.65
                         13.85         10.40         10.40
                         24.88         24.37         24.37
                          0.72          0.09          0.09
                          0.09          2.56          2.56
                          1.11          1.30          1.30
                          0.52          0.99          0.99

iter.            actual          best          current      stdev
-------------------------------------------------------------------------------

                   objf          objf          objf
   42/ 100       13.573        10.833        10.833
 nstrings: 1800
      obmean   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
     38.0576   0.00 DEfout   0.00 DEfout   0.00 DEfout    0.00
   online p.   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
    145.8004   0.00 DEfin    0.00 DEfin    0.00 DEfin     0.00
  offline p.   0.00 DWipc    0.00 DWipc    0.00 DWipc     0.00
     15.4972   0.00 DEipc    0.00 DEipc    0.00 DEipc     0.00
               0.00 DWhpc    0.00 DWhpc    0.00 DWhpc     0.00
               0.00 DEhpc    0.00 DEhpc    0.00 DEhpc     0.00
               0.00 DWhpt    0.00 DWhpt    0.00 DWhpt     0.00
               0.00 DEhpt    0.00 DEhpt    0.00 DEhpt     0.00
              -3.00 DWipt   -3.00 DWipt   -3.00 DWipt     0.22
               3.00 DEipt    2.85 DEipt    2.85 DEipt     0.18
               0.00 DWlpt    0.00 DWlpt    0.00 DWlpt     0.00
               0.00 DElpt    0.00 DElpt    0.00 DElpt     0.00
               0.00 DCa      0.00 DCa      0.00 DCa       0.00
 environment and power setting parameters
                520.6555      519.9066      519.9066
                 96.6327       96.5667       96.5667
                284.6523      284.7407      284.7407
 fault classes
    0    0    0    0    0 1800    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                                0.75          0.75
                                0.07          0.07
                                0.09          0.09
                                RMS           RMS
                                0.04          0.04
                                ABD           ABD
                                0.01          0.01
                          1.68          1.41          1.41
                          2.14          0.99          0.99
                          1.08          1.32          1.32
                          0.21          0.82          0.82
                          0.12          0.38          0.38
                          1.11          0.93          0.93
                          0.48          0.03          0.03
                          1.52          0.97          0.97
                          1.22          0.31          0.31
                          0.50          0.04          0.04
                         11.71         12.83         12.83
                         27.93         29.15         29.15
                          0.23          1.60          1.60
                          1.96          1.67          1.67
                          1.22          0.03          0.03
                          0.11          0.33          0.33
```

b. 1 faulty component, 5 biases

```
iter.        actual        best        current     stdev
-------------------------------------------------------------------------

                objf         objf         objf
  22/ 100      14.047       13.197       13.305
 nstrings: 3600
     obmean   0.00 DWfan   0.00 DWfan   0.00 DWfan    1.30
    89.1762   0.00 DEfout  0.00 DEfout  0.00 DEfout   0.90
  online p.  -3.00 DWfan  -2.99 DWfan  -2.99 DWfan    0.78
   177.5420   3.00 DEfin   2.99 DEfin   2.99 DEfin    0.52
 offline p.   0.00 DWipc   0.00 DWipc   0.00 DWipc    0.26
    21.6276   0.00 DEipc   0.00 DEipc   0.00 DEipc    0.19
              0.00 DWhpc   0.00 DWhpc   0.00 DWhpc    0.00
              0.00 DEhpc   0.00 DEhpc   0.00 DEhpc    0.00
              0.00 DWhpt   0.00 DWhpt   0.00 DWhpt    0.00
              0.00 DEhpt   0.00 DEhpt   0.00 DEhpt    0.00
              0.00 DWipt   0.00 DWipt   0.00 DWipt    0.00
              0.00 DEipt   0.00 DEipt   0.00 DEipt    0.00
              0.00 DWlpt   0.00 DWlpt   0.00 DWlpt    0.00
              0.00 DElpt   0.00 DElpt   0.00 DElpt    0.00
              0.00 DCa     0.00 DCa     0.00 DCa      0.41
 environment and power setting parameters
               478.6312     478.7163     478.6351
                93.4464      93.5768      93.5768
               256.9610     257.2748     257.3005
 fault classes
    0 1778    0    0    0    0    0    0 1332   17    0    0
    0    0    0   66    0    0    0    0  407    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                              0.09         0.00
                              0.13         0.13
                              0.31         0.34
                              RMS          RMS
                              0.00         0.00
                              ABD          ABD
                              0.00         0.00
                    5.52         5.52         5.56
                   12.74        12.14        12.27
                   13.35        12.34        12.27
                   11.54        10.65        10.78
                    8.14         7.09         7.01
                    2.17         2.16         2.21
                    0.79         1.36         1.22
                    1.63         0.56         0.49
                    0.98         0.88         1.08
                    1.55         2.77         2.85
                    0.62         0.16         0.30
                    0.49         0.71         0.80
                    1.23         0.85         0.60
                    1.54         0.95         0.93
                    2.76         2.78         2.83
                    0.28         0.00         0.01

iter.          actual        best        current     stdev
-------------------------------------------------------------------------

                objf         objf         objf
   9/ 100      14.978       14.303       14.303
 nstrings: 3600
     obmean  -3.00 DWfan  -2.98 DWfan  -2.98 DWfan    1.02
   125.6883   3.00 DEfout  2.95 DEfout  2.95 DEfout   0.93
  online p.   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.36
   268.8633   0.00 DEfin   0.00 DEfin   0.00 DEfin    0.20
 offline p.   0.00 DWipc   0.00 DWipc   0.00 DWipc    0.13
    20.0958   0.00 DEipc   0.00 DEipc   0.00 DEipc    0.14
              0.00 DWhpc   0.00 DWhpc   0.00 DWhpc    0.08
              0.00 DEhpc   0.00 DEhpc   0.00 DEhpc    0.09
              0.00 DWhpt   0.00 DWhpt   0.00 DWhpt    0.11
              0.00 DEhpt   0.00 DEhpt   0.00 DEhpt    0.09
              0.00 DWipt   0.00 DWipt   0.00 DWipt    0.10
              0.00 DEipt   0.00 DEipt   0.00 DEipt    0.09
              0.00 DWlpt   0.00 DWlpt   0.00 DWlpt    0.13
              0.00 DElpt   0.00 DElpt   0.00 DElpt    0.14
              0.00 DCa     0.00 DCa     0.00 DCa      0.29
 environment and power setting parameters
```

```
                521.1629      522.1146      522.1146
                 96.5834       96.6594       96.6594
                256.3757      256.2912      256.2912
 fault classes
 3348    5    2    1    0    1    3    4   63    6    2   15
   17    5   92    1    1    4    1    4    7    1    0    0
    3    5    0    0    1    2    0    1    0    1    1    3
 environment and power setting parameters' accommodation
                              0.95          0.95
                              0.08          0.08
                              0.08          0.08
                              RMS           RMS
                              0.01          0.01
                              ABD           ABD
                              0.01          0.01
                     4.19          3.66          3.66
                    11.70          9.83          9.83
                    11.10         11.26         11.26
                    11.24          9.35          9.35
                    10.98         11.13         11.13
                     0.27          0.24          0.24
                     6.51          8.62          8.62
                     0.13          0.24          0.24
                     1.29          0.96          0.96
                     0.97          1.06          1.06
                     3.17          1.04          1.04
                     1.56          1.70          1.70
                     1.78          4.53          4.53
                     0.17          0.18          0.18
                     0.13          0.08          0.08
                     1.33          0.61          0.61

iter.             actual        best         current      stdev
-------------------------------------------------------------------------

                   objf          objf          objf
    8/ 100       14.092        13.870        13.870
 nstrings: 3600
      obmean   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.36
    133.3304   0.00 DEfout   0.00 DEfout   0.00 DEfout    0.33
  online p.    0.00 DWfan    0.00 DWfan    0.00 DWfan     0.43
    284.1394   0.00 DEfin    0.00 DEfin    0.00 DEfin     0.49
 offline p.    0.00 DWipc    0.00 DWipc    0.00 DWipc     0.47
     22.1161   0.00 DEipc    0.00 DEipc    0.00 DEipc     0.44
               0.00 DWhpc    0.00 DWhpc    0.00 DWhpc     0.16
               0.00 DEhpc    0.00 DEhpc    0.00 DEhpc     0.12
               0.00 DWhpt    0.00 DWhpt    0.00 DWhpt     0.20
               0.00 DEhpt    0.00 DEhpt    0.00 DEhpt     0.14
               0.00 DWipt    0.00 DWipt    0.00 DWipt     0.16
               0.00 DEipt    0.00 DEipt    0.00 DEipt     0.16
              -3.00 DWlpt   -3.00 DWlpt   -3.00 DWlpt     1.15
               3.00 DElpt    2.95 DElpt    2.95 DElpt     0.90
               0.00 DCa      0.00 DCa      0.00 DCa       0.83
 environment and power setting parameters
                521.1780      520.2272      520.2272
                 93.4029       93.5438       93.5438
                303.0080      304.0438      304.0438
 fault classes
   14   24   19    2    2    3  214    1   21   32    1    2
    3   75    5   12    4    2    2  201    9    3   15    3
  124    5    0    2    7    3    0   15    1   19    1 2754
 environment and power setting parameters' accommodation
                              0.95          0.95
                              0.14          0.14
                              1.04          1.04
                              RMS           RMS
                              0.01          0.01
                              ABD           ABD
                              0.00          0.00
                     1.41          1.98          1.98
                     2.28          2.96          2.96
                     0.25          3.16          3.16
                     1.21          0.82          0.82
                     1.11          1.88          1.88
                     0.79          0.07          0.07
                     0.95          1.78          1.78
                     2.47          0.75          0.75
```

12.40	14.63	14.63
19.78	16.04	16.04
10.97	12.23	12.23
24.04	20.74	20.74
13.18	16.42	16.42
2.15	0.25	0.25
0.49	0.15	0.15
0.99	0.08	0.08

```
iter.        actual        best        current     stdev
-------------------------------------------------------------------------

                objf         objf         objf
  23/ 100      15.825       13.254       13.277
 nstrings: 3600
     obmean   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.25
    84.1583   0.00 DEfout  0.00 DEfout  0.00 DEfout   0.26
  online p.   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.09
   164.2029   0.00 DEfin   0.00 DEfin   0.00 DEfin    0.17
 offline p.   0.00 DWipc   0.00 DWipc   0.00 DWipc    0.15
    21.1388   0.00 DEipc   0.00 DEipc   0.00 DEipc    0.17
              0.00 DWhpc   0.00 DWhpc   0.00 DWhpc    0.12
              0.00 DEhpc   0.00 DEhpc   0.00 DEhpc    0.12
              0.00 DWhpt   0.00 DWhpt   0.00 DWhpt    0.05
              0.00 DEhpt   0.00 DEhpt   0.00 DEhpt    0.00
              0.00 DWipt   0.00 DWipt   0.00 DWipt    0.02
              0.00 DEipt   0.00 DEipt   0.00 DEipt    0.03
              0.00 DWlpt   0.00 DWlpt   0.00 DWlpt    0.15
              0.00 DElpt   0.00 DElpt   0.00 DElpt    0.43
              3.00 DCa     2.74 DCa     2.69 DCa      0.78
 environment and power setting parameters
               520.5807     519.8711     519.6559
                93.4649      93.7007      93.5804
               257.1578     257.7499     257.5669
 fault classes
   54    5   17    0    2    0  111 3130   12    1    0    0
    0    0  117    1    7    0    0    0    5    0    0    0
    0   61    0    0    0   28    0    0    0    0    1   48
 environment and power setting parameters' accommodation
                            0.71         0.92
                            0.24         0.12
                            0.59         0.41
                            RMS          RMS
                            0.07         0.08
                            ABD          ABD
                            0.02         0.02
                   1.04         0.74         0.77
                   1.32         0.95         0.04
                   0.16         1.25         0.78
                   2.30         1.45         2.02
                   4.83         3.07         3.60
                   3.62         3.40         3.37
                   0.29         0.11         0.35
                   0.33         1.39         0.79
                   8.62         9.74        10.09
                  23.65        22.03        22.80
                  11.21        13.21        14.28
                  28.90        29.14        30.38
                  11.85        13.87        14.05
                   0.16         0.46         0.02
                   0.49         0.16         0.41
                   1.29         0.29         1.12
```

c. 2 faulty components, 2 biases

```
iter.          actual        best        current     stdev
-------------------------------------------------------------------------

                objf         objf         objf
  34/ 100      18.609       19.479       22.934
 nstrings: 1800
     obmean  -3.00 DWfan  -2.78 DWfan  -2.33 DWfan    0.53
    84.4209   2.00 DEfout  1.60 DEfout  2.00 DEfout   0.62
  online p.   0.00 DWfan   0.00 DWfan   0.00 DWfan    0.00
   198.0701   0.00 DEfin   0.00 DEfin   0.00 DEfin    0.00
```

```
 offline p.    0.00 DWipc    0.00 DWipc    0.00 DWipc     0.00
     28.7028   0.00 DEipc    0.00 DEipc    0.00 DEipc     0.00
               0.00 DWhpc    0.00 DWhpc    0.00 DWhpc     0.00
               0.00 DEhpc    0.00 DEhpc    0.00 DEhpc     0.00
               0.00 DWhpt    0.00 DWhpt    0.00 DWhpt     0.00
               0.00 DEhpt    0.00 DEhpt    0.00 DEhpt     0.00
               0.00 DWipt    0.00 DWipt    0.00 DWipt     0.00
               0.00 DEipt    0.00 DEipt    0.00 DEipt     0.00
              -3.00 DWlpt   -2.99 DWlpt   -2.98 DWlpt     0.44
               1.00 DElpt    1.52 DElpt    1.84 DElpt     0.47
               0.00 DCa      0.00 DCa      0.00 DCa       0.00
 environment and power setting parameters
               478.3298      479.2324      480.1206
                93.4063       93.6927       93.5422
               285.3480      285.7388      285.3430
 fault classes
    0    0    0    0    0    0    0    0    0    0    0    0
    0 1800    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                           0.90          1.79
                           0.29          0.14
                           0.39          0.01
                           RMS           RMS
                           0.19          0.29
                           ABD           ABD
                           0.08          0.11
                    3.14          2.92          3.40
                    1.25          2.98          0.54
                    7.59          7.93          8.38
                    2.44          3.08          4.76
                    3.20          3.02          3.53
                    1.19          1.14          0.65
                    1.67          1.85          0.36
                    0.78          0.64          0.07
                    1.29          0.63          1.29
                    0.12          0.35          0.41
                    0.93          0.88          0.43
                   25.91         20.26         19.97
                    0.50          0.29          1.02
                    0.81          1.28          0.91
                    1.06          0.15          0.97
                    0.22          0.26          4.58

iter.         actual         best         current      stdev
-------------------------------------------------------------------------

                  objf          objf          objf
  100/ 100       11.016        10.982        11.503
 nstrings: 1800
     obmean    0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
     14.2292   0.00 DEfout   0.00 DEfout   0.00 DEfout    0.00
   online p.   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
     82.3419   0.00 DEfin    0.00 DEfin    0.00 DEfin     0.00
 offline p.    0.00 DWipc    0.00 DWipc    0.00 DWipc     0.00
     16.7340   0.00 DEipc    0.00 DEipc    0.00 DEipc     0.00
              -3.00 DWhpc   -2.71 DWhpc   -2.75 DWhpc     0.09
               2.00 DEhpc    1.83 DEhpc    1.86 DEhpc     0.06
               0.00 DWhpt    0.00 DWhpt    0.00 DWhpt     0.00
               0.00 DEhpt    0.00 DEhpt    0.00 DEhpt     0.00
              -3.00 DWipt   -2.81 DWipt   -2.82 DWipt     0.03
               2.00 DEipt    2.34 DEipt    2.30 DEipt     0.07
               0.00 DWlpt    0.00 DWlpt    0.00 DWlpt     0.00
               0.00 DElpt    0.00 DElpt    0.00 DElpt     0.00
               0.00 DCa      0.00 DCa      0.00 DCa       0.00
 environment and power setting parameters
               519.9454      519.8641      519.6795
                93.3548       93.3619       93.4030
               275.1387      275.0257      275.2144
 fault classes
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0 1800    0    0    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                           0.08          0.27
                           0.01          0.05
                           0.11          0.08
```

```
                                RMS             RMS
                                0.14            0.12
                                ABD             ABD
                                0.07            0.06
                        1.33            1.23            1.33
                        1.81            1.68            1.77
                        0.23            0.19            0.29
                        0.25            0.42            0.24
                        0.62            0.20            0.74
                        3.60            3.46            3.63
                        0.03            0.24            0.14
                        0.46            0.92            0.30
                        0.55            0.36            0.87
                       20.64           22.88           22.03
                        0.10            0.27            0.06
                       28.20           29.64           28.72
                        1.42            1.62            0.94
                        0.43            0.01            0.04
                        0.17            0.05            0.25
                        0.01            0.34            0.90

iter.        actual          best          current      stdev
-------------------------------------------------------------------------

                      objf            objf            objf
  100/ 100          15.995          10.902          11.220
 nstrings: 1800
      obmean    0.00 DWfan      0.00 DWfan      0.00 DWfan      0.00
     13.9211    0.00 DEfout     0.00 DEfout     0.00 DEfout     0.00
   online p.    0.00 DWfan      0.00 DWfan      0.00 DWfan      0.00
    102.4469    0.00 DEfin      0.00 DEfin      0.00 DEfin      0.00
  offline p.    3.00 DWipc      2.97 DWipc      2.96 DWipc      0.03
     22.5958    3.00 DEipc      2.96 DEipc      2.97 DEipc      0.08
                0.00 DWhpc      0.00 DWhpc      0.00 DWhpc      0.00
                0.00 DEhpc      0.00 DEhpc      0.00 DEhpc      0.00
                0.00 DWhpt      0.00 DWhpt      0.00 DWhpt      0.00
                0.00 DEhpt      0.00 DEhpt      0.00 DEhpt      0.00
               -3.00 DWipt     -2.99 DWipt     -2.99 DWipt      0.02
                2.00 DEipt      1.72 DEipt      1.72 DEipt      0.06
                0.00 DWlpt      0.00 DWlpt      0.00 DWlpt      0.00
                0.00 DElpt      0.00 DElpt      0.00 DElpt      0.00
                0.00 DCa        0.00 DCa        0.00 DCa        0.00
 environment and power setting parameters
                479.2954        478.0057        478.0862
                 96.5690         96.5860         96.5891
                284.8650        284.8148        284.8379
 fault classes
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0 1800
    0    0    0    0    0    0    0    0    0    0    0    0
 environment and power setting parameters' accommodation
                                1.29            1.21
                                0.02            0.02
                                0.05            0.03
                                RMS             RMS
                                0.08            0.07
                                ABD             ABD
                                0.03            0.03
                        1.32            1.36            1.35
                        0.88            0.03            0.12
                        0.62            0.13            0.24
                        0.20            0.77            0.68
                        3.10            2.67            2.77
                        2.37            2.49            2.51
                        0.83            0.03            0.11
                        0.17            0.04            0.06
                        0.08            0.41            0.30
                        1.30            2.32            2.17
                       15.72           16.53           16.43
                       26.47           32.15           31.88
                        2.03            0.34            0.48
                        1.44            0.10            0.23
                        0.52            0.05            0.15
                        1.12            0.15            0.05

iter.        actual          best          current      stdev
-------------------------------------------------------------------------
```

```
                objf          objf          objf
 100/ 100       15.561        22.267        24.015
nstrings: 1800
     obmean   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
     30.8810  0.00 DEfout   0.00 DEfout   0.00 DEfout    0.00
  online p.   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.00
    116.7637  0.00 DEfin    0.00 DEfin    0.00 DEfin     0.00
 offline p.   3.00 DWipc    2.83 DWipc    2.88 DWipc     0.08
     36.4328  3.00 DEipc    2.58 DEipc    2.29 DEipc     0.21
              0.00 DWhpc    0.00 DWhpc    0.00 DWhpc     0.00
              0.00 DEhpc    0.00 DEhpc    0.00 DEhpc     0.00
             -2.00 DWhpt   -2.06 DWhpt   -2.05 DWhpt     0.09
              3.00 DEhpt    2.87 DEhpt    2.78 DEhpt     0.07
              0.00 DWipt    0.00 DWipt    0.00 DWipt     0.00
              0.00 DEipt    0.00 DEipt    0.00 DEipt     0.00
              0.00 DWlpt    0.00 DWlpt    0.00 DWlpt     0.00
              0.00 DElpt    0.00 DElpt    0.00 DElpt     0.00
              0.00 DCa      0.00 DCa      0.00 DCa       0.00
environment and power setting parameters
             478.6794      477.7491      476.8656
              96.5644       96.6772       96.6979
             284.4451      284.7309      284.6165
fault classes
   0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0 1800    0
   0    0    0    0    0    0    0    0    0    0    0    0
environment and power setting parameters' accommodation
                          0.93          1.81
                          0.11          0.13
                          0.29          0.17
                          RMS           RMS
                          0.13          0.20
                          ABD           ABD
                          0.06          0.08
                  1.61          1.76          1.86
                  1.27          1.07          1.39
                  0.01          0.78          0.19
                  1.88          2.54          2.44
                  0.51          0.32          0.23
                  0.11          0.58          0.85
                 12.24         10.57         10.44
                 14.81         14.59         15.64
                  0.57          0.31          0.04
                  0.85          0.54          0.75
                  3.03          2.64          2.87
                  1.24          6.94         12.17
                  1.37          0.73          0.01
                  0.29          0.15          0.50
                  0.95          1.32          0.71
                  1.86          2.60          1.74
```

d. 2 faulty components, 5 biases

```
iter.        actual        best        current     stdev
-------------------------------------------------------------------------

                objf          objf          objf
  51/ 100       14.161        9.615         9.648
 nstrings: 1200
     obmean   0.00 DWfan    0.00 DWfan    0.00 DWfan     0.07
     30.5020  0.00 DEfout   0.00 DEfout   0.00 DEfout    0.02
   online p. -2.00 DWfan   -1.91 DWfan   -1.88 DWfan     0.24
    134.7900  1.00 DEfin    1.18 DEfin    1.22 DEfin     0.21
  offline p.  0.00 DWipc    0.00 DWipc    0.00 DWipc     0.00
     26.4796  0.00 DEipc    0.00 DEipc    0.00 DEipc     0.00
              0.00 DWhpc    0.00 DWhpc    0.00 DWhpc     0.00
              0.00 DEhpc    0.00 DEhpc    0.00 DEhpc     0.00
              0.00 DWhpt    0.00 DWhpt    0.00 DWhpt     0.00
              0.00 DEhpt    0.00 DEhpt    0.00 DEhpt     0.00
             -3.00 DWipt   -2.99 DWipt   -2.98 DWipt     0.13
              3.00 DEipt    2.96 DEipt    2.90 DEipt     0.20
              0.00 DWlpt    0.00 DWlpt    0.00 DWlpt     0.00
              0.00 DElpt    0.00 DElpt    0.00 DElpt     0.00
              0.00 DCa      0.00 DCa      0.00 DCa       0.00
 environment and power setting parameters
             479.6776      479.3790      479.0369
```

```
                96.5389       96.7914       96.7436
               255.7784      255.9694      255.8711
fault classes
   0    0    0    0    0    0    0    0    0    0    0    0
   1    0    0    0    0    0 1199    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0
environment and power setting parameters' accommodation
                                0.30          0.64
                                0.25          0.20
                                0.19          0.09
                                RMS           RMS
                                0.06          0.07
                                ABD           ABD
                                0.02          0.03
                      1.99          1.90          1.97
                      1.29          0.42          0.99
                      2.07          1.94          2.35
                     10.04         10.41         11.18
                     10.86         10.61         10.99
                      0.26          0.14          0.07
                      1.08          0.91          0.55
                      0.19          0.15          0.16
                      0.71          0.19          0.27
                     24.48         24.18         24.68
                     13.24         12.73         13.29
                     32.45         32.11         33.30
                      3.24          1.82          1.28
                      0.43          0.38          0.01
                      0.78          1.08          1.33
                      2.12          0.69          0.65

iter.        actual        best        current     stdev
-------------------------------------------------------------------------

                 objf          objf          objf
 100/ 100        6.897         5.831         5.868
nstrings: 1800
     obmean   0.00 DWfan    0.00 DWfan    0.00 DWfan    0.00
      7.0765  0.00 DEfout   0.00 DEfout   0.00 DEfout   0.00
  online p.   0.00 DWfan    0.00 DWfan    0.00 DWfan    0.00
     70.4018  0.00 DEfin    0.00 DEfin    0.00 DEfin    0.00
 offline p.   0.00 DWipc    0.00 DWipc    0.00 DWipc    0.00
     14.6802  0.00 DEipc    0.00 DEipc    0.00 DEipc    0.00
              0.00 DWhpc    0.00 DWhpc    0.00 DWhpc    0.00
              0.00 DEhpc    0.00 DEhpc    0.00 DEhpc    0.00
              0.00 DWhpt    0.00 DWhpt    0.00 DWhpt    0.00
              0.00 DEhpt    0.00 DEhpt    0.00 DEhpt    0.00
             -2.00 DWipt   -2.07 DWipt   -2.04 DWipt    0.01
              3.00 DEipt    2.70 DEipt    2.74 DEipt    0.03
              0.00 DWlpt    0.00 DWlpt    0.00 DWlpt    0.00
              0.00 DElpt    0.00 DElpt    0.00 DElpt    0.00
              2.00 DCa      2.00 DCa      2.05 DCa      0.04
environment and power setting parameters
                478.9559      478.0532      478.3465
                 96.6201       96.6365       96.6451
                256.2094      256.0677      256.0660
fault classes
   0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0 1800    0
environment and power setting parameters' accommodation
                                0.90          0.61
                                0.02          0.02
                                0.14          0.14
                                RMS           RMS
                                0.08          0.07
                                ABD           ABD
                                0.03          0.02
                      2.05          2.11          2.09
                      0.56          0.08          0.50
                      0.85          0.04          0.13
                      0.12          0.39          0.22
                      0.71          1.47          1.43
                      0.18          0.01          0.05
                      0.35          0.85          0.27
                      0.44          0.35          0.21
                      9.71          9.85          9.50
```

22.79	23.76	23.68
12.98	13.41	12.74
27.45	32.29	31.40
9.98	10.95	10.48
0.10	0.02	0.24
0.08	0.42	0.65
1.47	0.08	0.09

iter.	actual	best	current	stdev
	objf	objf	objf	
54/ 100	6.658	7.744	9.370	

nstrings: 1200

obmean	-2.00	DWfan	-1.95	DWfan	-2.03	DWfan	0.13
24.7687	0.30	DEfout	0.42	DEfout	0.70	DEfout	0.27
online p.	0.00	DWfan	0.00	DWfan	0.00	DWfan	0.00
124.6973	0.00	DEfin	0.00	DEfin	0.00	DEfin	0.00
offline p.	0.00	DWipc	0.00	DWipc	0.00	DWipc	0.00
22.6381	0.00	DEipc	0.00	DEipc	0.00	DEipc	0.00
	0.00	DWhpc	0.00	DWhpc	0.00	DWhpc	0.00
	0.00	DEhpc	0.00	DEhpc	0.00	DEhpc	0.00
	-3.00	DWhpt	-2.99	DWhpt	-2.96	DWhpt	0.17
	2.00	DEhpt	1.89	DEhpt	2.08	DEhpt	0.12
	0.00	DWipt	0.00	DWipt	0.00	DWipt	0.00
	0.00	DEipt	0.00	DEipt	0.00	DEipt	0.00
	0.00	DWlpt	0.00	DWlpt	0.00	DWlpt	0.00
	0.00	DElpt	0.00	DElpt	0.00	DElpt	0.00
	0.00	DCa	0.00	DCa	0.00	DCa	0.00

environment and power setting parameters

521.1058	520.9013	521.7197
96.6142	96.6156	96.2773
255.8283	255.6593	255.3124

fault classes

0	0	0	0	0	0	0	0	0	0	0	1200
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

environment and power setting parameters' accommodation

0.20	0.61
0.00	0.34
0.17	0.52
RMS	RMS
0.04	0.11
ABD	ABD
0.02	0.04

9.88	9.81	10.40
10.85	11.15	13.56
11.86	12.24	12.18
0.38	1.21	0.22
0.34	0.81	1.28
0.35	0.69	0.48
0.24	0.07	0.95
0.54	1.32	1.21
1.24	0.32	0.21
1.65	1.29	0.89
10.51	10.42	11.90
28.37	31.69	28.65
0.84	0.97	1.06
0.42	0.67	1.38
0.30	0.28	1.02
0.36	0.13	0.67

iter.	actual	best	current	stdev
	objf	objf	objf	
39/ 100	10.459	7.508	7.864	

nstrings: 1800

obmean	0.00	DWfan	0.00	DWfan	0.00	DWfan	0.00
40.2477	0.00	DEfout	0.00	DEfout	0.00	DEfout	0.00
online p.	0.00	DWfan	0.00	DWfan	0.00	DWfan	0.18
139.4514	0.00	DEfin	0.00	DEfin	0.00	DEfin	0.06
offline p.	0.00	DWipc	0.00	DWipc	0.00	DWipc	0.00
19.9487	0.00	DEipc	0.00	DEipc	0.00	DEipc	0.00
	0.00	DWhpc	0.00	DWhpc	0.00	DWhpc	0.00
	0.00	DEhpc	0.00	DEhpc	0.00	DEhpc	0.00

```
                0.00 DWhpt    0.00 DWhpt    0.00 DWhpt     0.00
                0.00 DEhpt    0.00 DEhpt    0.00 DEhpt     0.00
                0.00 DWipt    0.00 DWipt    0.00 DWipt     0.00
                0.00 DEipt    0.00 DEipt    0.00 DEipt     0.00
               -3.00 DWlpt   -2.81 DWlpt   -2.83 DWlpt     0.32
                3.00 DElpt    3.00 DElpt    2.62 DElpt     0.31
                2.00 DCa      2.40 DCa      2.65 DCa       0.35
environment and power setting parameters
                 478.7839      480.4653      479.9627
                  96.5900       96.4469       96.4524
                 256.4974      256.4594      256.4226
fault classes
   0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0   12    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0 1788
environment and power setting parameters' accommodation
                              1.68          1.18
                              0.14          0.14
                              0.04          0.07
                              RMS           RMS
                              0.12          0.21
                              ABD           ABD
                              0.04          0.09
                      7.77          8.32          8.35
                     10.42          8.41          6.68
                      0.93          1.86          2.21
                      1.07          1.08          0.22
                      0.97          0.39          0.20
                      0.39          0.55          0.39
                      1.66          0.38          1.56
                      2.01          1.68          1.47
                      0.12          0.53          0.71
                      0.95          0.50          0.28
                     11.79          8.30          6.28
                     27.19         19.83         23.21
                     12.46          9.57          8.50
                      1.48          0.39          0.26
                      0.42          0.10          0.42
                      0.43          0.05          0.14
```